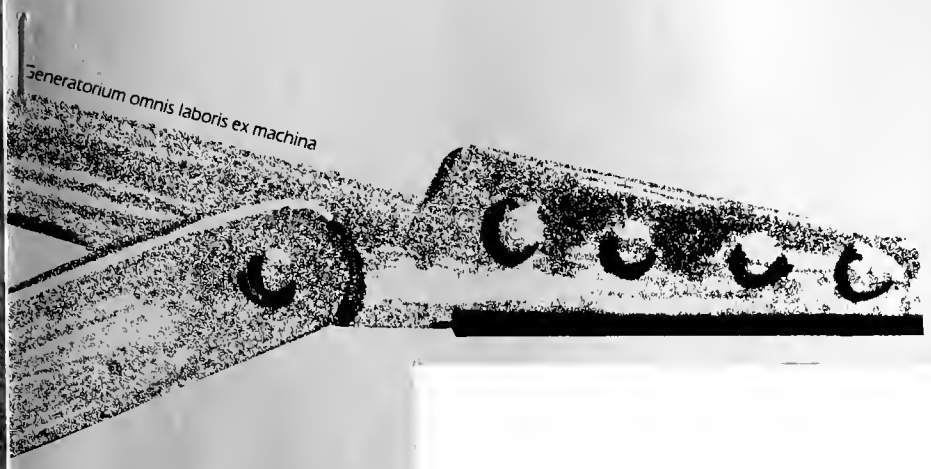
Robotics Research Technical Report



2 Piece Jig-Saw Puzzle Robot Assembly
with Vision, Position, and Force Feedback

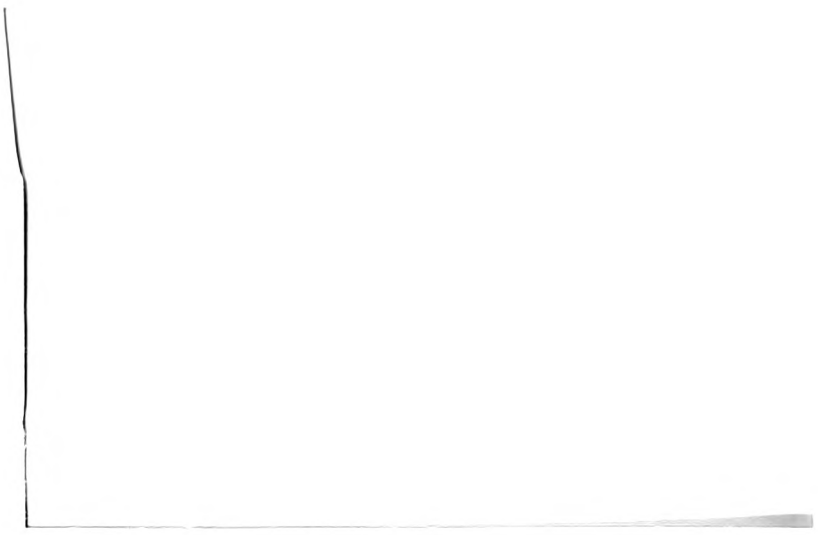
by

Grigore C. Burdea

Technical Report No. 290
Robotics Report No. 107
April, 1987

New York University
Institute of Mathematical Sciences

Computer Science Division
251 Mercer Street New York, N.Y. 10012



2 Piece Jig-Saw Puzzle Robot Assembly with Vision, Position, and Force Feedback

by

Grigore C. Burdea

Technical Report No. 290
Robotics Report No. 107
April, 1987

New York University
Dept. of Computer Science
Courant Institute of Mathematical Sciences
251 Mercer Street
New York, New York 10012

Work on this paper has been supported by Office of Naval Research Grant N00014-82-K-0381, National Science Foundation CER Grant DCR-83-20085, and by grants from the Digital Equipment Corporation and the IBM Corporation.

2 Piece Jig-Saw Puzzle Robot Assembly with Vision, Position and Force Feedback

Grigore C. Burdea[1]

Courant Institute of Mathematical Sciences
New York University
251 Mercer Street
New York, N.Y. 10012

ABSTRACT

This paper presents a robotic assembly of 2 3-D jig-saw puzzle pieces, using vision, position and force feedback. Geometrical aspects are discussed together with force history of the experiments. An installation using 7565 I.B.M. robot with a CCD camera on the robot arm is described. Successful experiments proved the possibility of assembling parts with complex contours when force feedback is used during fine assembly. A videotape of the demonstration is also available.

[1]Affiliated with Department of Applied Science, New York University

2 Piece Jig-Saw Puzzle Robot Assembly with Vision, Position and Force Feedback[2]

Grigore C. Burdea[3]

1. Introduction

Automated assembly of variable and complex shaped parts with relatively small tolerances is a substantially more demanding task than the *peg in a hole* problem treated in literature. Inoue [6] Simmunovic[10] Seltzer[9] have discussed the force aspects of peg in a hole insertion, and demonstrated successful mating with small tolerances. Their approach was to consider the geometry of parts as known, and use complex force sensors (such as the *Instrumented Center of Compliance*) to accommodate positioning errors during assembly. Further work by Lozano-Perez[1] ,[8] demonstrated the automated synthesis of fine motion strategies for circular peg in hole insertion. Buckley [3] extends the automated synthesis to rectangular pegs in rectangular holes. The experimental part of his thesis uses a similar I.B.M. 7565 robot as the one used by us, while the tolerances were also of the same order of magnitude (0.03 inches). Buckley's experiments were more general than the circular peg approach, since no axial symmetry was present. However in his motion synthesis Buckley simplifies the general problem in that he does not address rotational errors during assembly. This errors do usually develop and are crucial for parts with asymmetrical contours such as rectangular pegs.

The present work addresses aspects related to robot assembly of *variable shaped parts with no axial symmetry* such as *jig-saw puzzles*. Due to their complex geometry, rotational errors cannot be overlooked and are addressed

---

[2]Work on this project has been supported by Office of Naval Research Grant N00014-82-K-0381, National Science Foundation CER Grant DCR-83-20085, and by grants from the Digital Equipment Corporation and from I.B.M. Corporation.

[3]Affiliated with Department of Applied Science, New York University.

during the fine assembly part of our experiment. We chose to use puzzle pieces since Kalvin, Schonberg and Wolfson [7] ,[12] developed algorithms for simulated jig-saw puzzle assembly using vision data. They chose to use only the contour of the puzzle pieces rather than the actual picture presented by the puzzle, which led to the idea of attempting a robotic verification of Wolfson's algorithm, which sometimes may give erroneous solutions[4]

This paper describes techniques developed for automatic assembly of two puzzle pieces. Future work will extend this results to robotic assembly of a whole jig-saw puzzle. The outline of this report is as follows:

Chapter 2 describes the experimental installation, sensor calibration issues as well as puzzle piece modification for robot assembly.

Chapter 3 presents vision to robot transformations used for part detection and pick up.

Chapter 4 develops the geometrical transformations used during robot assembly. Robot moves are structured in *raw positioning, transition to fine assembly and fine assembly moves.*

Chapter 5 presents force readings during successful experiments as they relate to the type of robot moves analyzed in Chapter 4.

Chapter 6 determines acceptable level of errors for successful assembly. Here both geometric and sensorial implications are addressed and theoretical predictions are compared with experimental results.

The Appendix contains some of the software developed for vision and force assembly.

2. Experimental Installation

2.1. Description

The experimental system as presented in Fig. 1, 2 and 3, consists of an I.B.M. 7565 robot, a vision system, force sensors and mechanical force amplifier. The vision system uses an existing Fairchild CCD camera, with a 256x256 sensor array and a VICOM image processor with 512X512 array. The Fairchild camera sensor head is installed directly on the robot arm, at a fixed height above the robot work table. This eliminates the need for special lense drives to compensate for variable focal lengths, while allowing a complete scan of the table. The robot work area was covered with white photographic paper, and the wooden puzzles were covered with black nonreflective paper in order to boost the vision image quality.

A VAX minicomputer supervises the vision process and communicates with a SERIES 1 minicomputer which is the robot controller. Force sensors on the robot gripper allow for force feedback during assembly. In an earlier design [5] we included a special vacuum tool for parts handling. This was not finally implemented due to unstable grasp, slip and potential piece loss when

subject to assembly torques. This tool was replaced with a combination of handle on the piece and specially designed mechanical force amplifier as presented in Fig. 3. The criteria that led to the final gripper *nails* design were:

(1) stable four point grip on cylindrical handle;

(2) adaptive grip during initial part pickup (allowing the use of *center grasping*);

(3) no interference with existing *part presence sensor* on the gripper;

(4) mechanical force amplification.

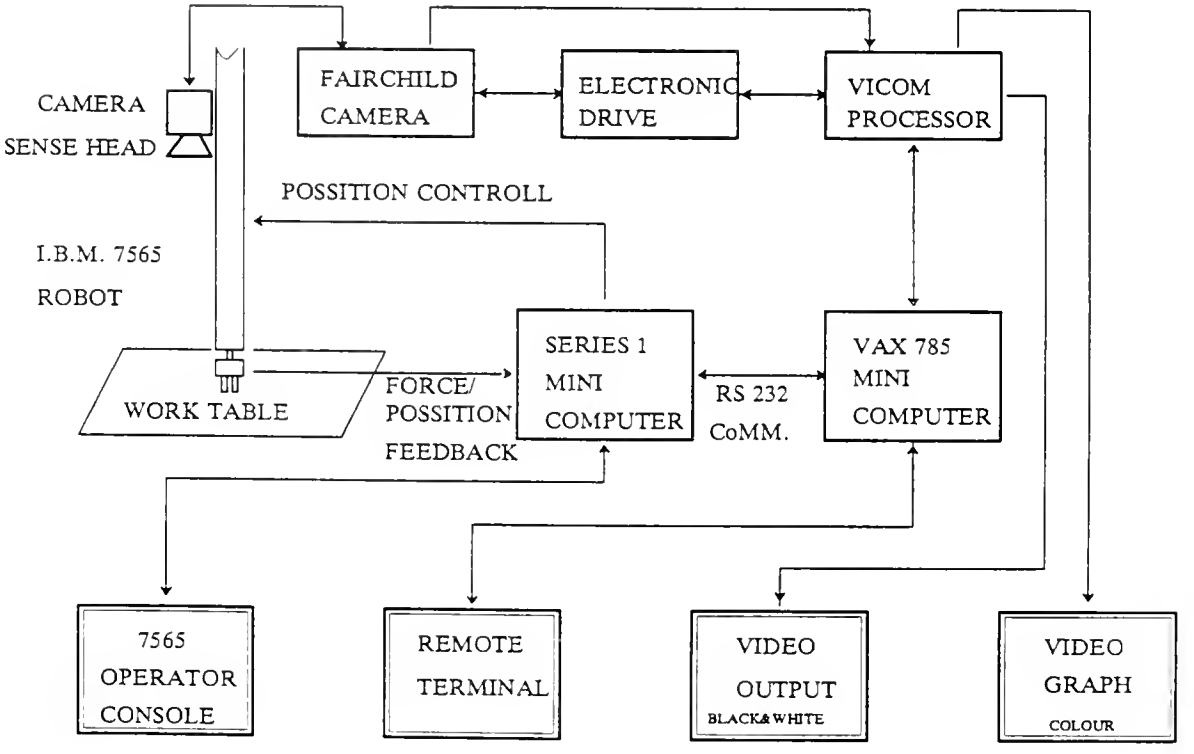


Figure 1 Experimental installation for puzzle assembly

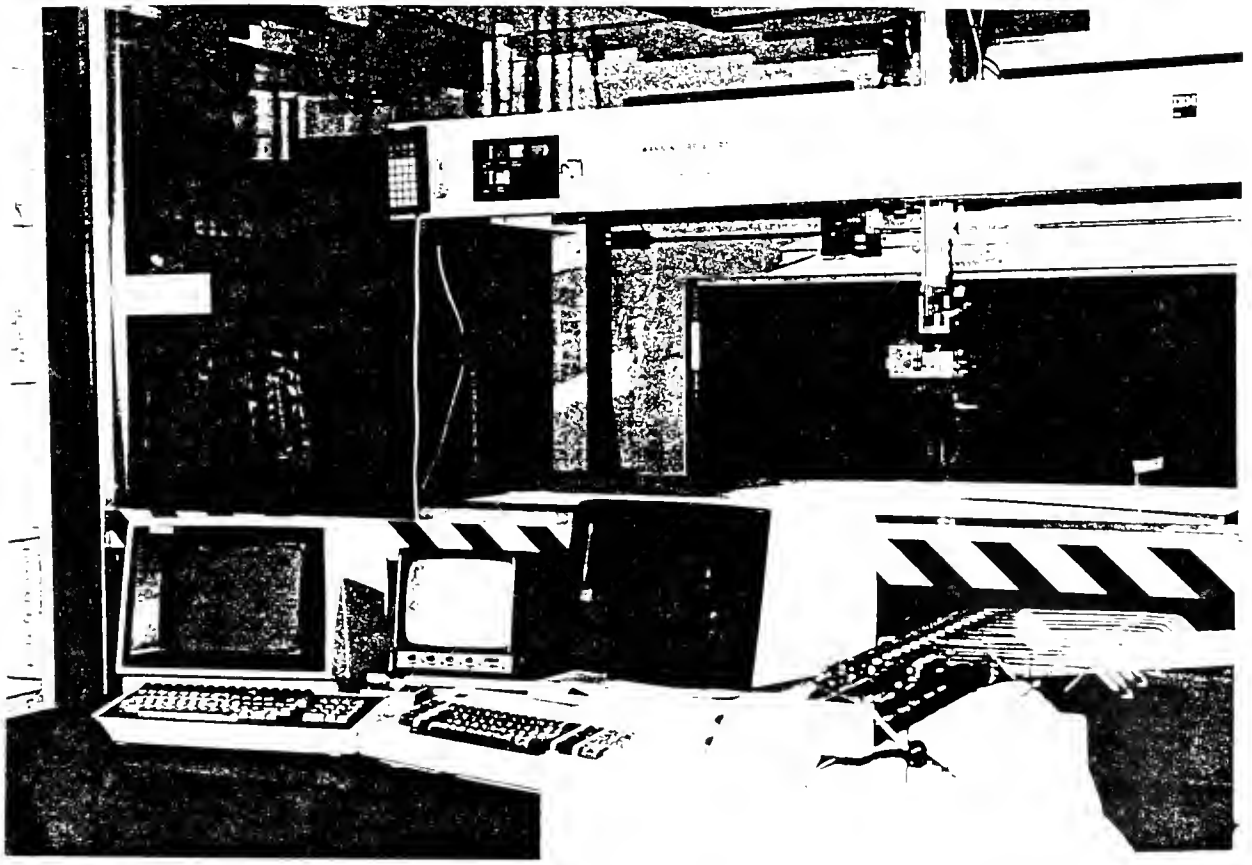
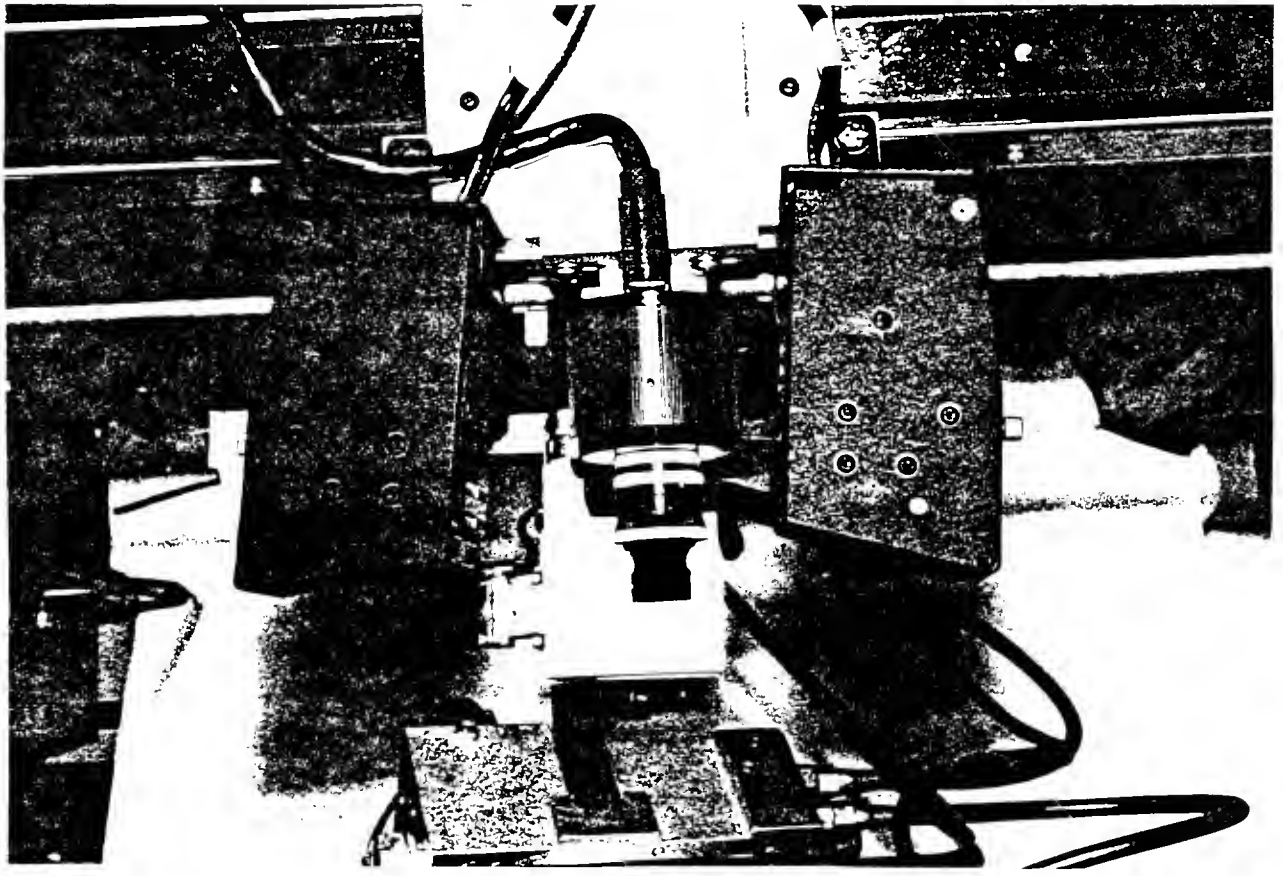
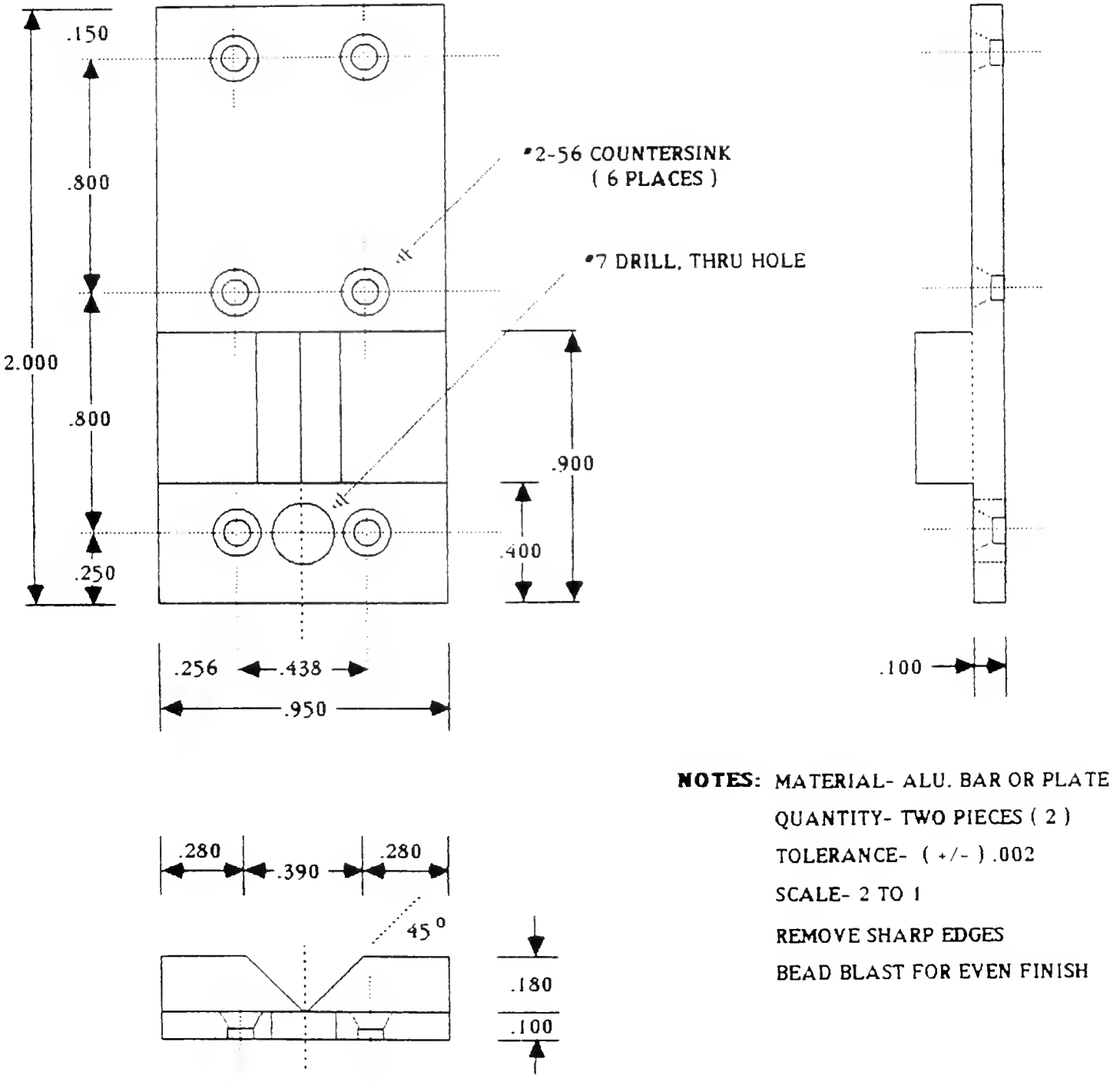


Fig. 2 IBM 7565 robot with vision system (above)
Fig. 3 CCD camera on robot arm (detail)



JOB: GRIPPER (RS-2)



NOTES: MATERIAL- ALU. BAR OR PLATE
QUANTITY- TWO PIECES (2)
TOLERANCE- (+/-) .002
SCALE- 2 TO 1
REMOVE SHARP EDGES
BEAD BLAST FOR EVEN FINISH

Fig. 4 Special gripper nails for 7565 I.B.M. robot

2.2. Piece preparation

For our experiment we selected an *of the shelf* wooden puzzle presented in Fig. 5. The vision matching algorithm uses only the piece shape, therefore we covered the piece picture with black nonreflective paper. This was necessary in order to boost the quality of vision images. It is important that the pieces do not slip on the work table during robot assembly. Therefore the bottom of the puzzle pieces was fitted with *Koroseal flexible magnetic strip*[2] in friction that prevents sliding at small assembly forces. The magnitude of the magnetic force determines the force envelope that the robot uses during assembly. This forces were determined experimentally to be about 1.4 N/sq. inch or about 3 N for puzzle pieces with a surface of 2 to 3 sq. inches. The robot gripper configuration (2 parallel fingers) does not allow direct grasping of the puzzle pieces. It was therefore necessary to mount a cylindrical handle on one of the puzzle pieces corresponding to its center of gravity. In this way the robot grasps on the handle when picking up the puzzle piece. Grasping has to be done without pushing or overturning the puzzle piece. The maximum grasping force used in *center grasp* is given by (2.1).

$$F_{grasp} = \frac{Ml}{h} \tag{2.1}$$

F_{grasp} = maximum grasping force used for center grasping
M = total magnetic pull
l = distance from the center of gravity to the puzzle edge
h = grasping force application point versus work table

$$M = k_{mag} A \tag{2.2}$$

k_{mag} = specific magnetic pulling force
A = puzzle piece area

The grasping force application point h depends on the handle length as well as the arm vertical coordinate and the gripper length as expressed in (2.3).

$$h = \frac{H + Z - Z_{table} - l_{gripper}}{2} \tag{2.3}$$

H = handle height above work table
Z = robot arm Z coordinate
Z_{table} = Z coordinate of work table surface

$l_{gripper}$ = robot gripper length

With (2.2) and (2.3) equation (2.1) becomes (2.4).

$$F_{grasp} = \frac{2k_{mag}Al}{H+Z-l_{gripper}-Z_{table}} \tag{2.4}$$

Using the physical parameters of the materials used, equation (2.4) becomes (2.5)

$$F_{grasp} = \frac{l}{0.213+0.066Z} \tag{2.5}$$

which yields grasping forces of about 1.2 N, without pushing the piece.

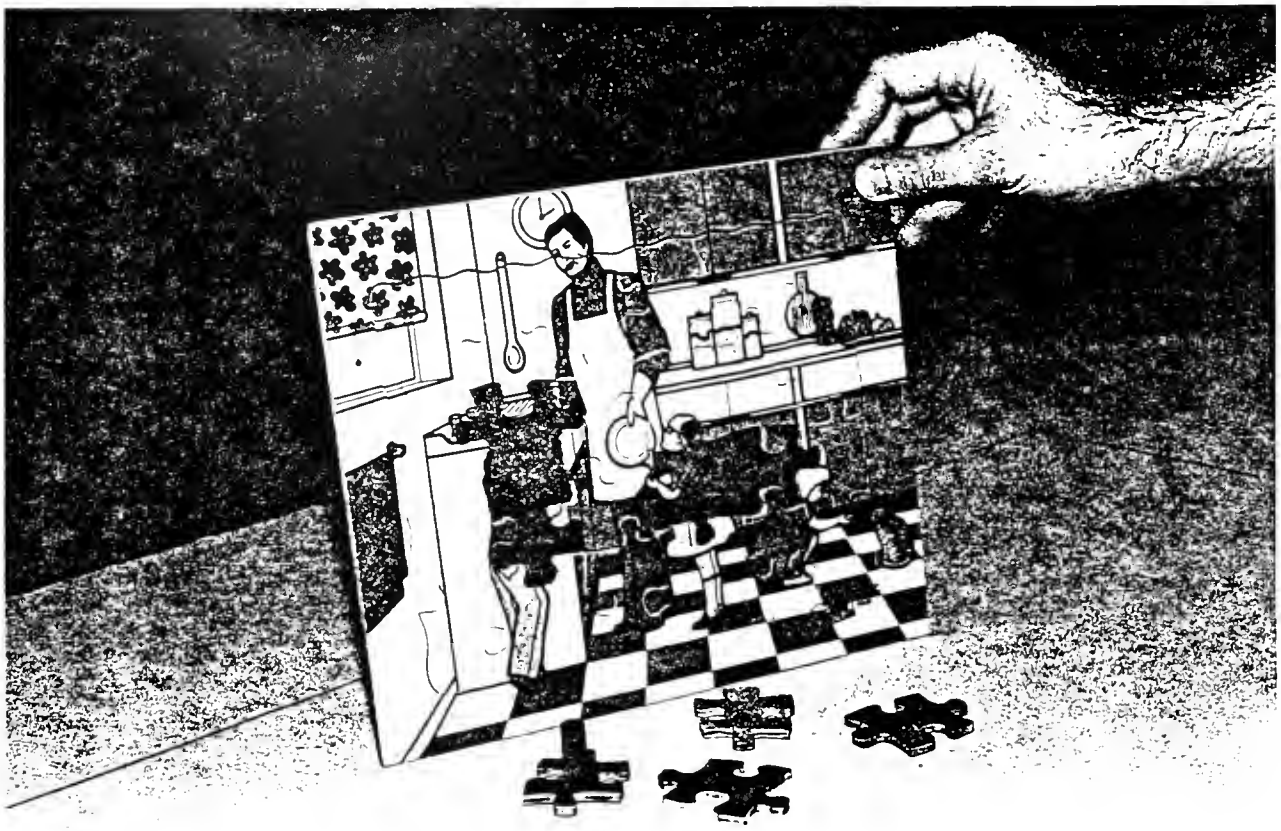
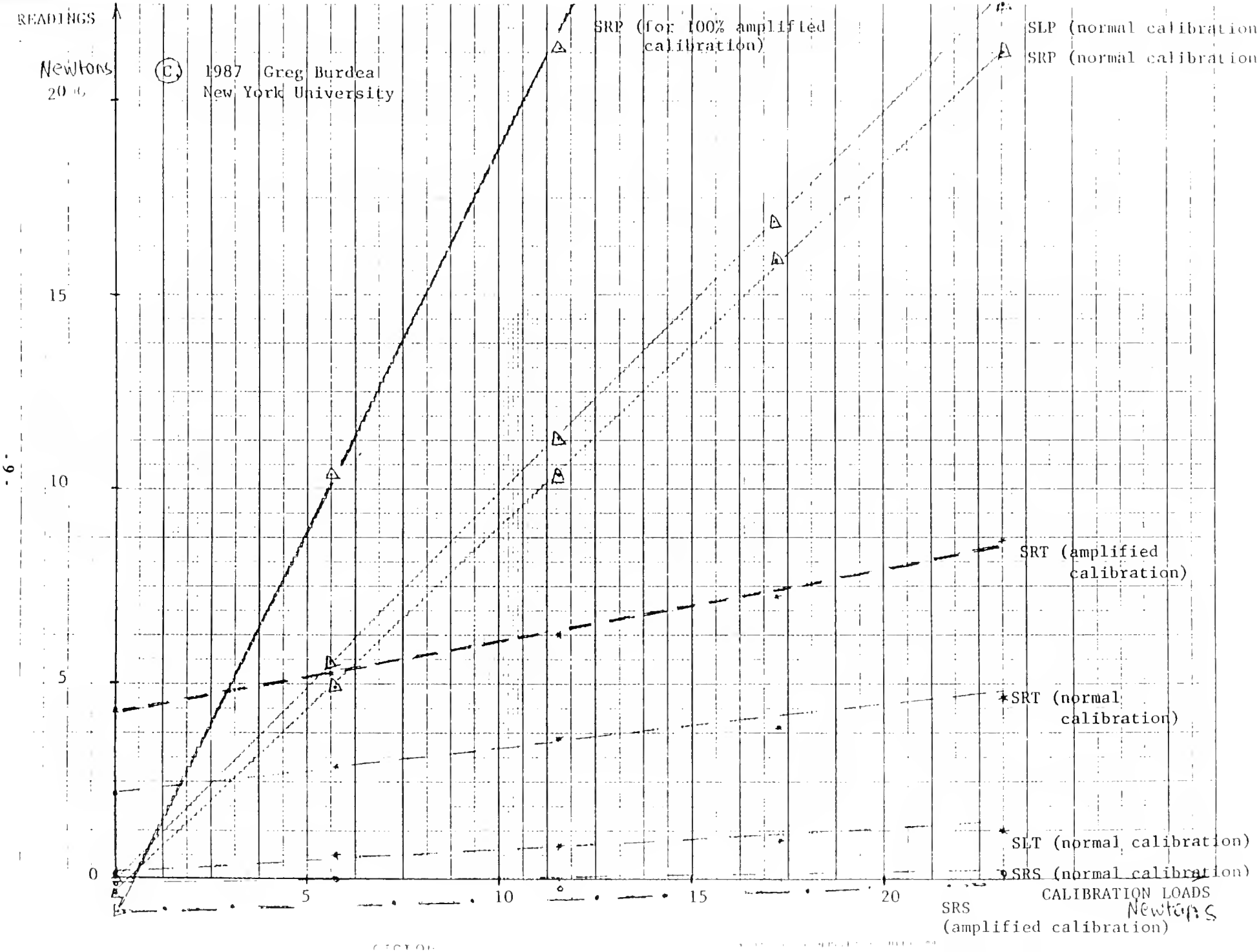


Fig. 5 Wooden jig-saw puzzle pieces used for assembly


READINGS
Newtons
© 1987 Greg Burdea
New York University
SRP (for 100% amplified calibration)
SLP (normal calibration
SRP (normal calibration
SRT (amplified calibration)
SRT (normal calibration)
SLT (normal calibration)
SRS (normal calibration)
CALIBRATION LOADS
Newtons
SRS (amplified calibration)
0
5
10
15
20

2.3. Force sensor recalibration

In section 2.2 we determined that horizontal assembly forces should not exceed 3 N in order to avoid pushing the puzzle pieces over robot work table. This condition is even more restrictive for piece pick up when the grasping force should be less than 2 N. The 7565 I.B.M. robot gripper is equipped with force sensors that provide 3D force information. This sensors are factory designed for a range of 0 to 100 N, with low resolution in the range of forces used during puzzle assembly. We have therefore taken several steps to amplify signal in the low senor range. One step is to modify the sensor calibration lookup table thus obtaining 100% amplification for small forces. A special table was created during a one time off-line calibration, and the low level software loaded in the system library. Since no high level software arithmetics is involved, there was no negative impact on the overall run time of the application. Fig. 6 presents sensor readings before and after recalibration, showing a 100% amplification for small forces. The adverse result of this procedure was an increase in signal noise (or uncertainty) from 0.15 to 0.30 N. The next step to increase readings was to add specially designed *nails* to the gripper. These nails, which are presented in Fig. 4, work as mechanical force amplifiers producing 50% pitch and side force amplification.

3. Vision steps for assembly

The vision steps required by piece detection, feature extraction and visual match are schematically presented in Fig. 7. The convention was to name the stationary puzzle piece as *Piece 1* while the piece that is picked up by the robot arm was named *Piece 2*.

3.1. Camera to robot coordinate transformations

For both visual search and piece pickup it is necessary to establish the correspondence between image and robot spaces. This relation is determined by a number of parameters such as *camera position offset* versus the robot arm, *camera x/y distortion factor* as well as *camera projection factor*. We define the camera position offset as the distance from the image center to the x/y robot link in robot coordinates. Here z coordinate is fixed and no rotation offset is taken into account. The x/y distortion factor is specific to each CCD camera and represents the number of pixels that corresponds to a given length along x and y axis. The projection factor depends on the lens used as well as camera to work table distance. It represents the number of inches that correspond to one pixel in the image. Taking all above factors into account, camera to robot coordinate transformations are expressed by (3.1) and (3.2).

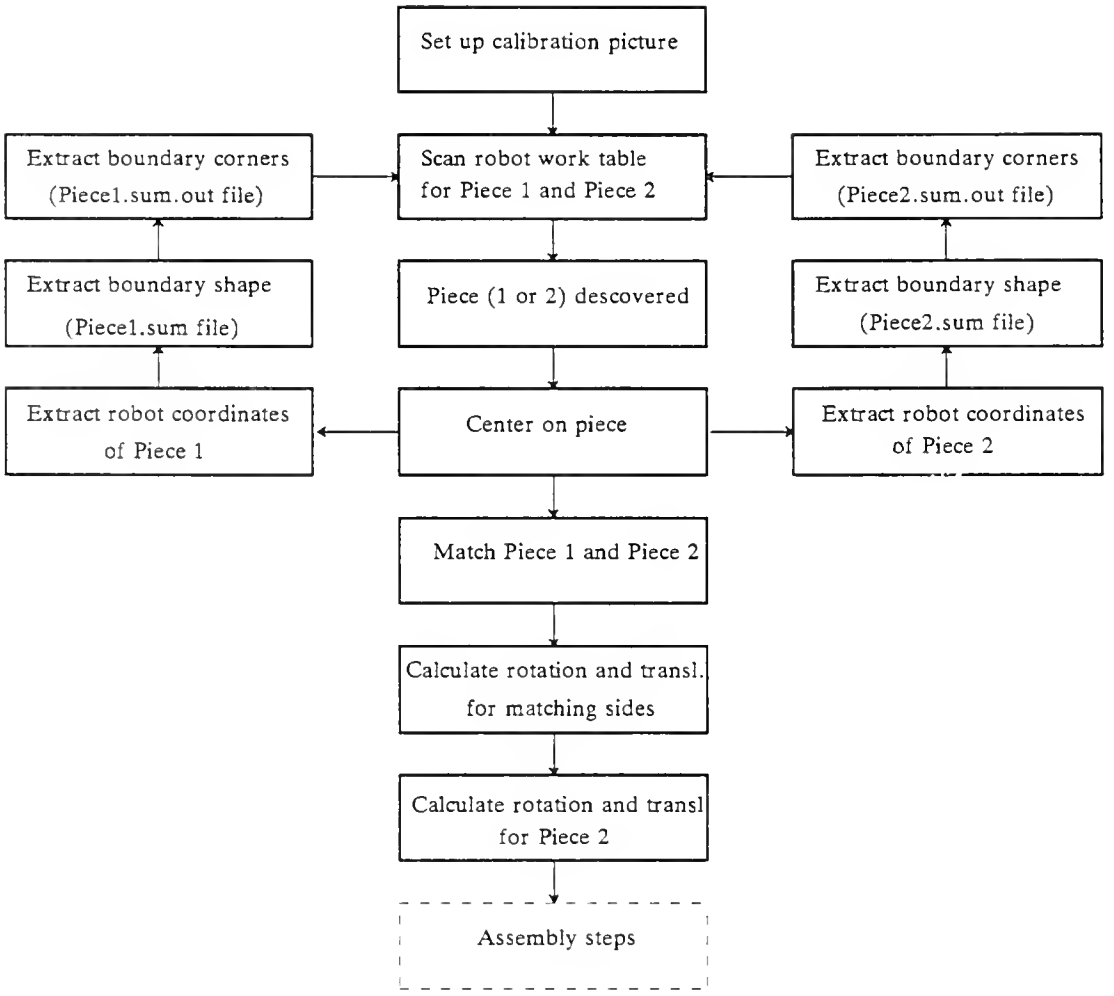


Fig. 7 Vision steps for assembly

$$X_{robot} = (256 - X_{vicom}) * PROJECTION - X_{offset} + X \quad (3.1)$$

$$Y_{robot} = \frac{(Y_{vicom} - 240) * PROJECTION}{XSCALE} - Y_{offset} + Y \quad (3.2)$$

X_{vicom}, Y_{vicom} = point VICOM coordinates
X_{robot}, Y_{robot} = point robot coordinates
$PROJECTION$ = projection factor
X_{offset} = camera offset in robot x direction
Y_{offset} = camera offset in robot y direction
$XSCALE$= camera distortion factor
X, Y = robot arm coordinates

3.2. Work table visual scan

As a matter of generality the puzzle pieces are placed randomly on the work table, and their location is not known to the assembly algorithm. There are however limitations on the puzzle piece position which has to be within the intersection of camera field of view and robot work envelope. Let us define $X_{robmin}, Y_{robmin}, X_{robmax}, Y_{robmax}$ as the minimum and maximum robot X,Y coordinates. Similarly $X_{cammln}, Y_{cammin}, X_{cammax}, Y_{cammax}$ are the camera field of view minim and maxim X,Y coordinates. Combining vision and robot assembly means that for a Cartesian robot such as 7565, puzzle piece coordinates have to satisfy (3.3) and (3.4).

$$X_{puzzle} = (X_{cammin}, X_{cammax}) \cap (X_{robmin}, X_{robmax}) \quad (3.3)$$

$$Y_{puzzle} = (Y_{cammin}, Y_{cammax}) \cap (Y_{robmin}, Y_{robmax}) \quad (3.4)$$

The restrictions (3.3), (3.4) limit the space that is scanned by the camera/robot assembly in order to discover the puzzle pieces. This *scanning space* is subdivided in *quadrants* which overlap so that the low quality border side of images can be discarded. During this initial assembly stage , we are interested to determine whether the puzzle pieces exist rather than extracting their shape. The visual scan therefore uses *compressed* images that give a significant speedup of processing. Each quadrant image is subsequently analyzed to detect part presence. If a part exists, ANALYZE returns its raw center of gravity in vision coordinates. The vision system is a 2D system that takes images of a 3D scene. This produces vision errors if pieces are not centered under the camera. The robot has therefore the additional task of centering the camera on top of a discovered piece in order to minimize vision errors. This is done by the routine CENTER.

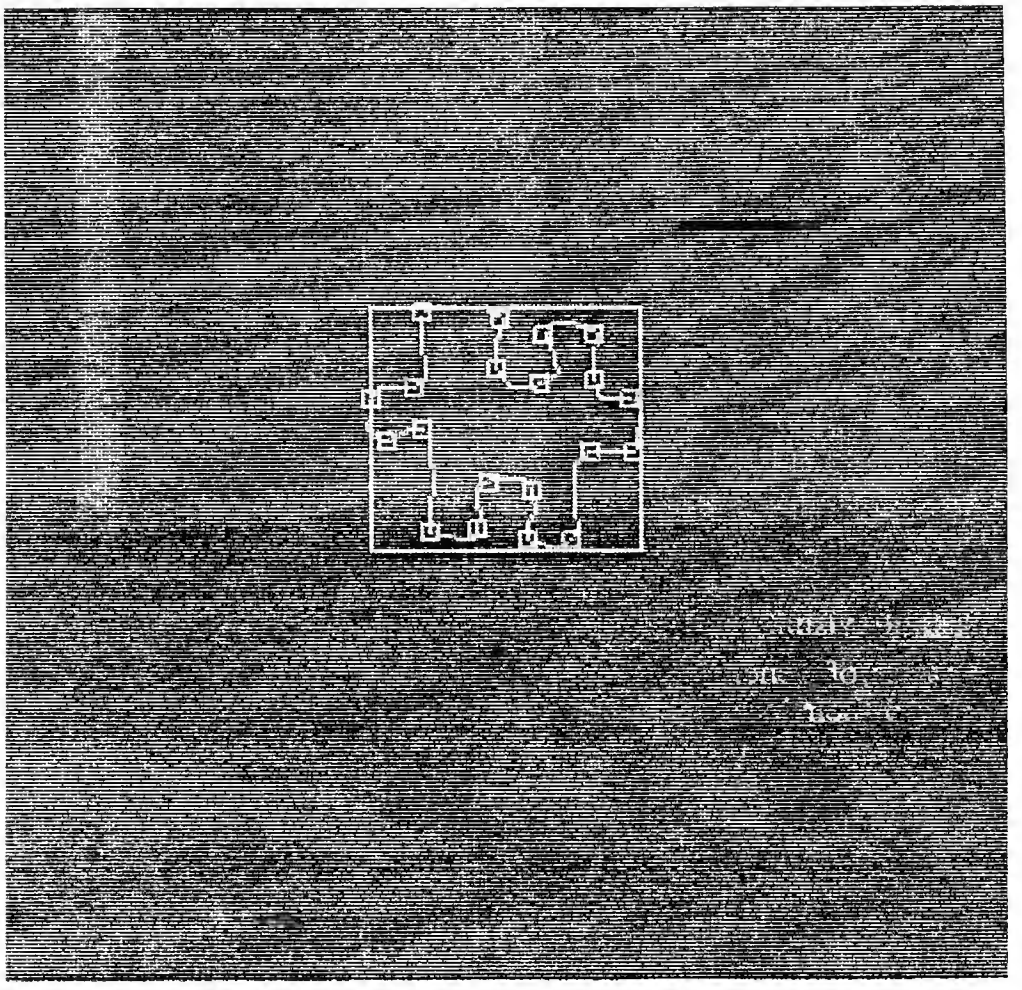


Fig. 8 CORNER output

3.3. Shape extraction and visual match

Once the camera is centered on the puzzle piece, its shape is extracted using SHAPE and CORNER routines (Edith Schonberg). The output is a sampled curve divided into four sides as presented in Fig. 8. This becomes input for MATCH routine (A. Kalvin) that returns a score for matching sides of two puzzle pieces. Our algorithm selects the smallest of 16 match scores, corresponding to the 2 sides that most likely match. MATCH also returns the rotation and translation of the matching sides. An automated rotation and translation is done using the handle on the puzzle piece, rather than the matching sides. The target for center of gravity of the *rotated* puzzle piece is obtained from MATCH output using (3.5), (3.6).

$$G_{3x} = \frac{P_{2x} + LEFTX\,(XSCALE - 1)}{XSCALE} + sin\,(\theta)(G_{1y} - P_{1y}) + \\ + \frac{cos\,(\theta)(G_{1x} - P_{1x})}{XSCALE} \tag{3.5}$$

$$G_{3y} = P_{2y} + \frac{sin\,(\theta)(P_{1x} - G_{1x})}{XSCALE} + cos\,(\theta)(G_{1y} - P_{1y}) \tag{3.6}$$

$G_{3x,y}$ = center of gravity of rotating piece after rotation
$G_{1x,y}$ = center of gravity of rotating piece
$P_{1x,y}$ = center of gravity of matching side on rotating piece
$P_{2x,y}$ = center of gravity of matching side on stationary piece
θ = rotation angle of matching side (given by MATCH)
$LEFTX$ = leftmost X_{vicom} coordinate of stationary piece
$XSCALE$ = camera distortion factor

4. Robot assembly steps

The robot assembly moves are presented schematically in Fig. 9. We have adopted Simmunovic's[10] notation when considering the over all automated assembly as divided in three steps: raw positioning, transition to fine assembly and fine assembly. During *raw positioning* Piece 2 is picked up, rotated and translated to Piece 2, tilted and lowered on top of Piece 1. The raw positioning task is to bring Piece 2 in contact with Piece 1 in such a way as to allow further assembly moves to eliminate both rotation and translation errors. This leads to a purposely induced *overshoot* of Piece 2 so that the relative position of the two pieces is known. During *transition moves* the two pieces remain in contact while their relative position changes to a *nested* configuration. It is during transition moves that translational errors are eliminated. The final and crucial step is *fine assembly* based on force feedback. During these moves Piece 2 is pushed into Piece 1 while rotational errors are eliminated. Finally Piece 2 is tilted back to horizontal configuration without contact loss with Piece 1. This represents the last step in fine assembly. The following subsections give more details regarding the assembly moves, while Chapter 5 depicts force sensor history for actual assemblys.

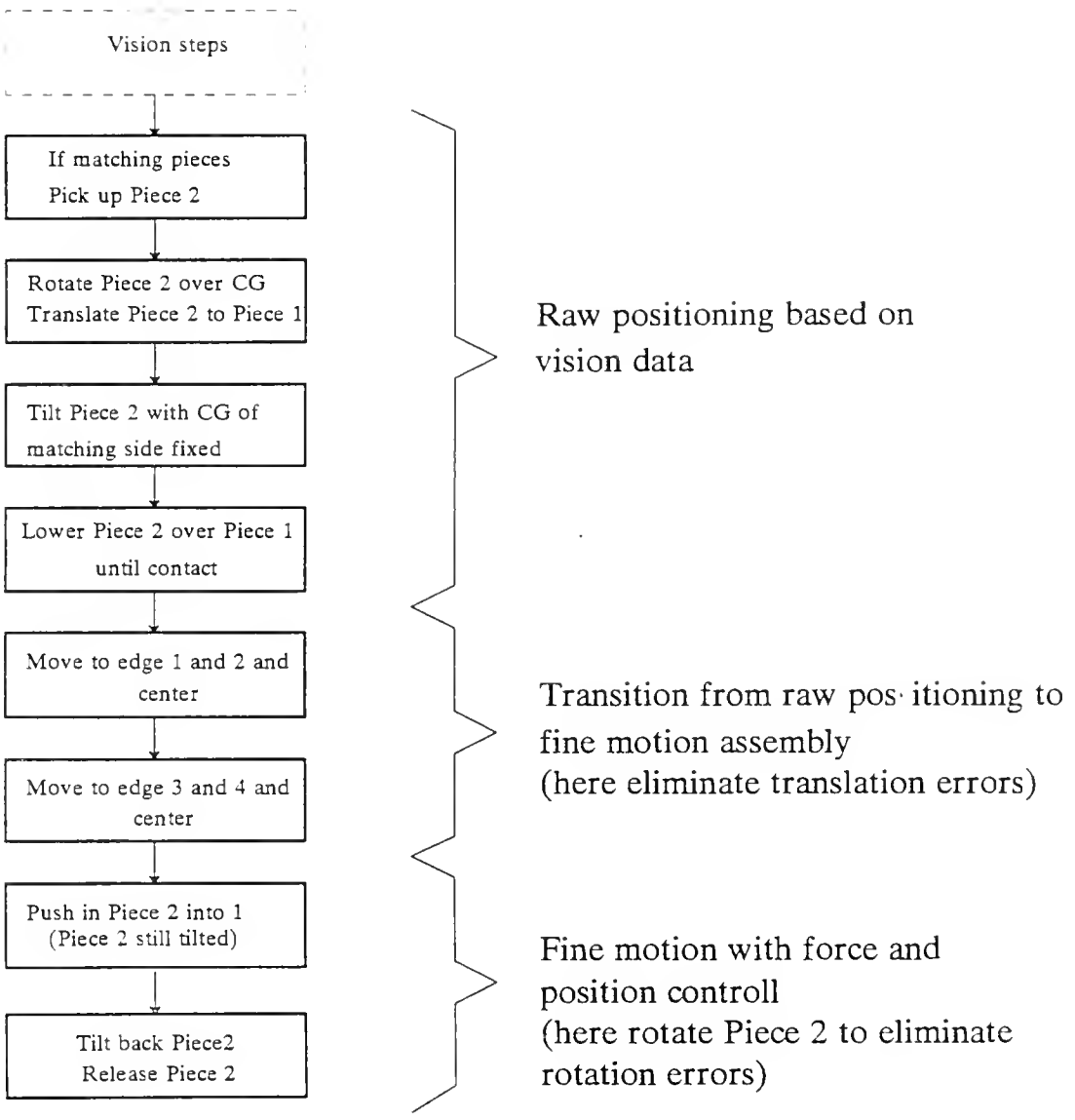


Fig. 9 Robot assembly steps

4.1. Raw positioning based on vision data

4.1.1. Approaching move in free space

The robot goal for initial approach of Piece 1 after picking up Piece 2 is given by (4.1) and (4.2).

$$G_{3xrobot} = (256 - G_{3x})PROJECTION - X_{offset} + X_1 \tag{4.1}$$

$$G_{3yrobot} = \frac{(G_{3y} - 240)PROJECTION}{XSCALE} - Y_{offset} + Y_1 \tag{4.2}$$

$G_{3xrobot,\ 3yrobot}$ = robot target for approach
$G_{3x,y}$ = center of gravity of moving piece after rotation given by (3.5-6)
X_1, Y_1 = robot arm coordinates when stationary piece was in center of image

The Z arm coordinates during this move are determined by *collision avoidance* criteria. Since no obstacles are present, this becomes a free space move. APPROACH calculates X,Y coordinates for this move and is included in Appendix.

4.1.2. Tilting move

The piece that is picked up by robot has a *convexity* that has to get into the *concavity* of the stationary piece. Generaly speaking, the convexity becomes an *asymmetric peg* and the concavity an *asymmetric hole*. The measured average tolerance for assembly (0.03 inches) is the same order of magnitude as camera projection factor. It is therefore necessary to tilt the piece in order to encrease tolerances. After tilting tolerances are about the same as the concavity diameter. The geometry of the two puzzle pieces determine the actual tilting angle. For our case this angle is 10 degrees. Let us define a Cartesian *Object System of Coordinates* as presented in Fig. 10. The origin of this system of coordinates is the center of gravity of the side with convexity (peg) P_1, while Y_{obj} passes through G_1. During tilting P_1 is maintained at fixed X,Y,Z robot coordinates. Object to robot coordinate transformations are given by equations (4.3), (4.4) and (4.5).

$$X = X_{robot} + sin(\beta)(a - Y_{obj}) + \cos(\beta)X_{obj} \tag{4.3}$$

$$Y = Y_{robot} + \sin(\beta)X_{obj} + cos(\beta)(Y_{obj} - a) \tag{4.4}$$

$$Z = Z_{robot} + Z_{obj} - f - b \tag{4.5}$$

X,Y,Z = point in robot system of coordinates
$X_{robot},Y_{robot},Z_{robot}$=robot arm coordinates in robot system of coordinates
X_{obj},Y_{obj},Z_{obj}= point coordinates in object system of coordinates
β = rotation angle between robot and object coordinate systems
a = distance from P_1 to G_1
b = handle length
f = distance from handle top to robot arm X,Y,Z link
θ= tilting angle

Robot target for tilting the piece thus becomes (4.6) to (4.9).

$$X_{tilt}=X_{robot}+sin(\beta)(a-acos(\theta)+(b+f)sin(\theta)) \quad (4.6)$$

$$Y_{tilt}=Y_{robot}-cos(\beta)(a-acos(\theta)+(b+f)sin(\theta)) \quad (4.7)$$

$$Z_{tilt}=Z_{robot}+(b+f)(cos(\theta)-1)+asin(\theta) \quad (4.8)$$

$$Pitch=\theta \quad (4.9)$$

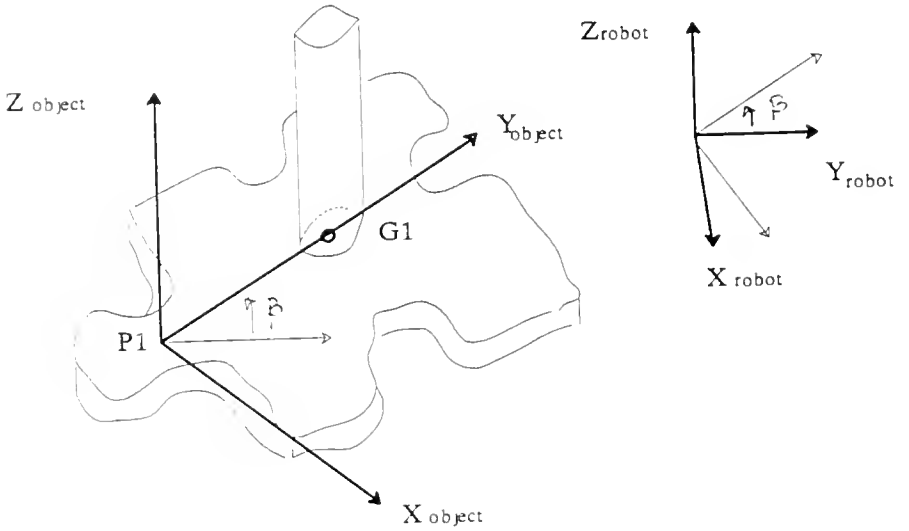


Fig. 10 Object system of coordinates

4.1.3. Bringing the pieces in contact

This is the last move during raw positioning assembly. Here the free space assumption does not apply, since the move terminates upon contact with Piece 1. This is a different termination condition than for the previous moves, and requires force feedback. Will and Grossman[11] call this type of moves *guarded moves*. Buckley[3] gives the relation between position errors and force feedback in what he names the *generalized spring* equation (4.10).

$$x = x_c + \frac{f_r}{k} \tag{4.10}$$

x = robot actual position
x_c = robot commanded position
f_r = sensed reaction force
k = system stiffness constant

Equation (4.10) may also be written as (4.11) that gives reaction forces as a function of position errors that occur when the robot arm moves against an obstacle (in our case the stationary piece).

$$f_r = \frac{x - x_c}{k} \tag{4.11}$$

Equation (4.11) is the termination condition for all transition and fine assembly moves described in subsequent chapters.

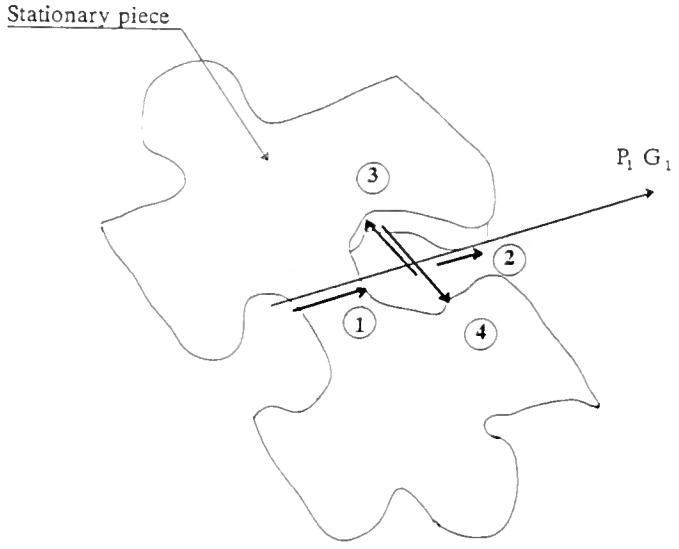


Fig. 11 Transition moves

4.2. Transition moves

These moves are done in order to *nest* the convexity on Piece 2 into the concavity of Piece 1. The first transition move is a sliding on top of Piece 1 that terminates when contact is lost at edge 1, as presented in Fig. 11. The sliding is done along the *generalized direction* $P_1\ G_1$. After edge 1 was detected, the next move is towards edge 2, where contact between the two pieces is reestablished. An average of the two termination points represents the target for next move. The same procedure is repeated for edge 3 and 4, but the move is done on a trajectory *perpendicular* to P_1G_1. Averaging the four contact points eliminates translation errors that were induced by the vision system.

4.3. Fine assembly moves

After completion of transition moves Piece 2 is nested with Piece 1, but on top of it and tilted. During fine assembly Piece 2 is to be pushed down into Piece 1 and then tilted back to a horizontal position. The task for tilt back is to maintain contact with Piece 1 and rotate vertically around the contact point. It is therefore necessary to calculate the distance JG_1 as shown in Fig. 13. Let l be the length of JG_1.

$$l = \frac{Z_{table} - (f+b+t)cos(\theta) - t}{sin(\theta)} \tag{4.12}$$

l = distance from the contact point to Piece 2 center of gravity
t= piece thickness
θ = tilting angle

4.3.1. Push in move with rotation errors

First the convexity of Piece 2 is brought in contact with Piece 1 by moving Piece 2 back to edge 1 along G_1P_1. Once the contact established, the next move is a pushing of Piece 2 into Piece 1 with the target given by (4.13) to (4.16).

$$X_{pushin} = X_{robot} - tsin(\beta)sin(\theta) \tag{4.13}$$

$$Y_{pushin} = Y_{robot} + tcos(\beta)sin(\theta) \tag{4.14}$$

$$Z_{pushin} = Z_{robot} - tcos(\theta) \tag{4.15}$$

$$Pitch = \theta \tag{4.16}$$

The tilting angle θ is such that if rotation errors exist, the *elbow* of Piece 2 hits the top of Piece 1 before the pushing in completes. The algorithm has to distinguish between the two possible cases of *right or left elbow hit*. The force-torque system that corresponds to these cases is presented if Fig. 14.

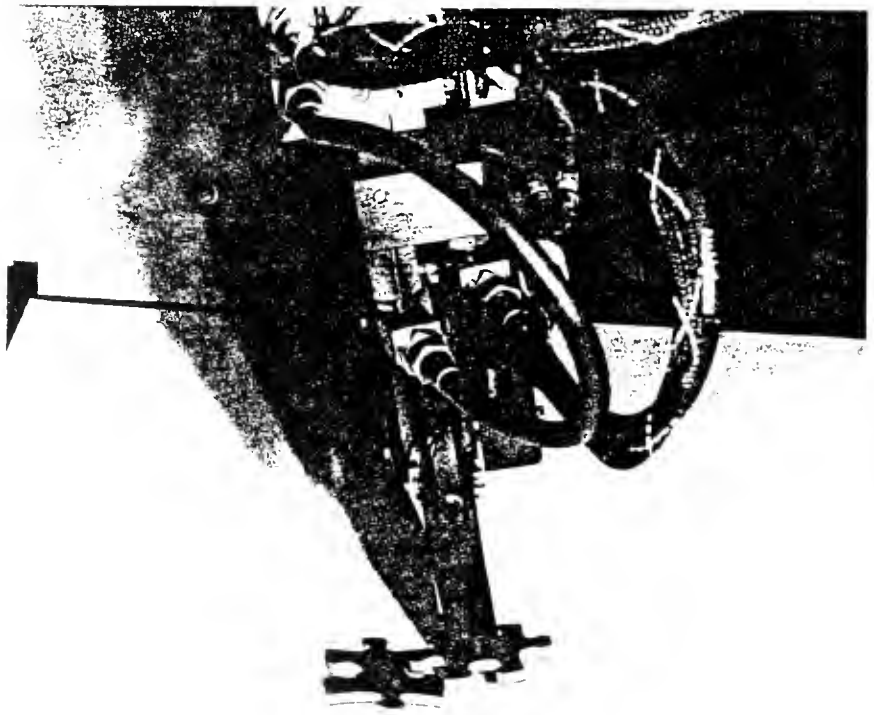


Fig. 12 Relative possition after transition moves

In order to eliminate rotation errors, Piece 2 is rotated by an angle γ, and the goal for this move is (4.17) to (4.20).

$$X_{rotation} = X_{robot} + (l\cos(\theta) - (f+b)\sin(\theta))$$
$$(\sin(\gamma+\beta) + \sin(\beta)) \qquad (4.17)$$

$$Y_{rotation} = Y_{robot} + (l\cos(\theta) - (f+b)\sin(\theta))$$
$$(\cos(\gamma+\beta) - \cos(\beta)) \qquad (4.18)$$

$$Z_{rotation} = Z_{robot} \qquad (4.19)$$

$$Jaw_{rotation} = Jaw_{robot} + \gamma \qquad (4.20)$$

$X, Y, Z, Jaw_{rotation}$ = target for rotation move
γ = rotation angle around contact point J

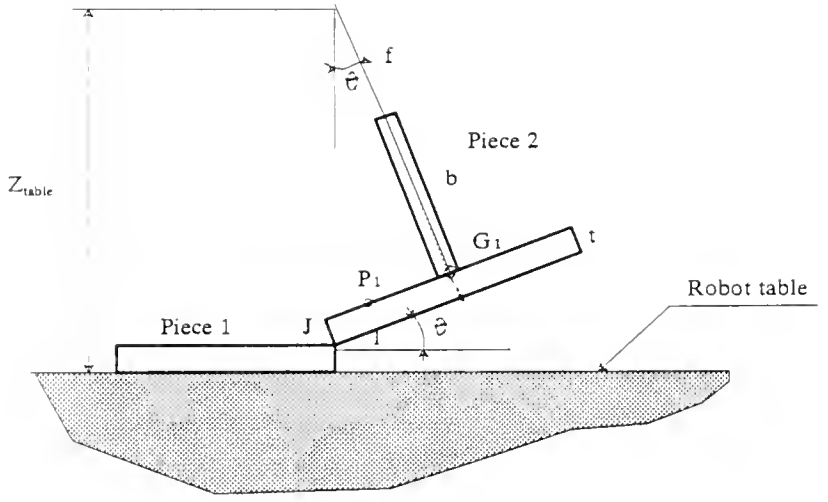


Fig. 13 Notation for JG1 calculation

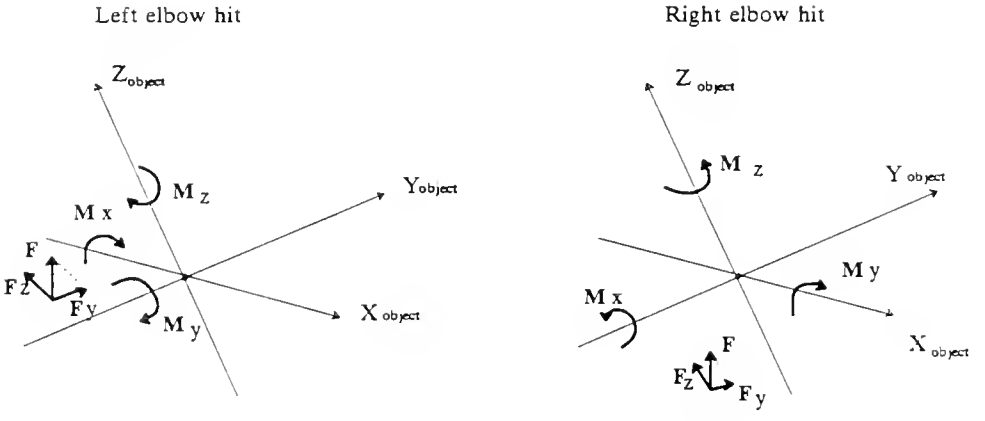


Fig. 14 Force-torque system for elbow hit

The rotation move terminates when contact between the two pieces is lost which results in a modification of force readings.

4.3.2. Tilt back

This is the last move of fine assembly which brings Piece 2 to a horizontal position without loss of Piece 1. During this move the target is given by (4.21) to (4.24).

$$X_{tiltback} = X_{robot} + sin(\beta+\gamma)(l - lcos(\theta) - (f+b)sin(\theta)) \quad (4.21)$$

$$Y_{tiltback} = Y_{robot} - cos(\beta+\gamma)(l - lcos(\theta) - (f+b)sin(\theta)) \quad (4.22)$$

$$Z_{tiltback} = Z_{robot} + (f+b)(cos(\theta) - 1) - lsin(\theta) \quad (2.23)$$

$$Pitch_{tiltback} = Pitch_{robot} - \theta \quad (4.24)$$

Fig. 15 presents the pieces position during tilt back move.

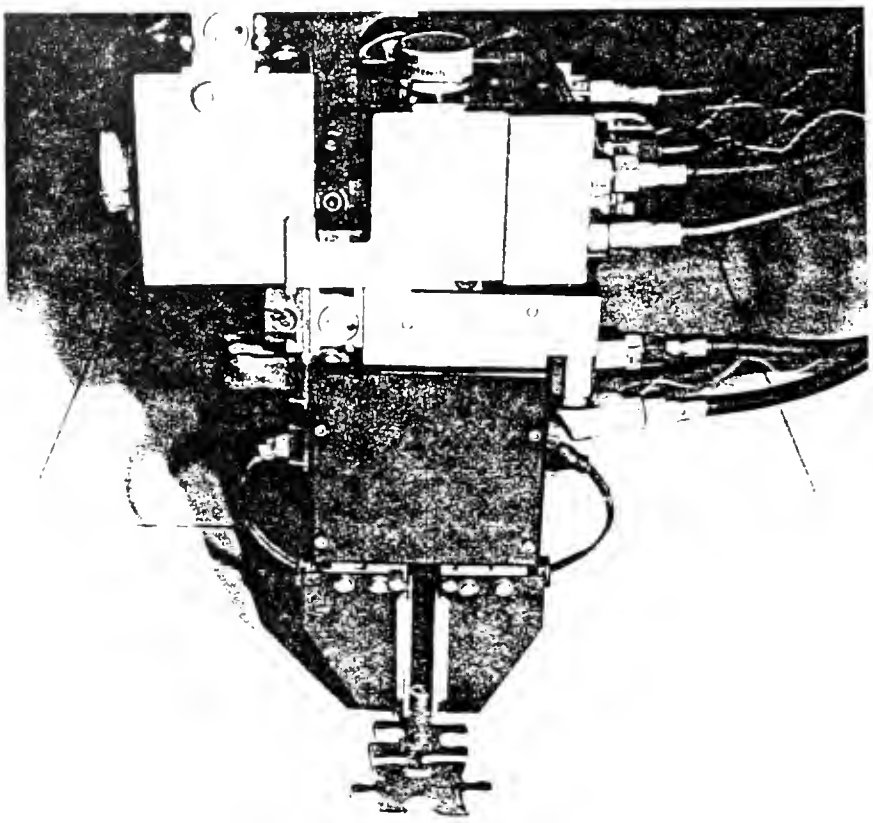


Fig. 15 Tilt back move

5. Force history

IBM 7565 force sensor geometry is presented in Fig. 16. Each gripper finger has *tip, side and pitch strain gages* giving force readings in three directions. Sensor readings are influenced by a number of factors such as *orientation, sensor offset, calibration, mechanical amplification, crosstalk, vibrations.*

A simple test was conducted in order to determine the effect of gripper orientation on force sensor output. The gripper executed a full rotation from -135 to 135 degrees, while maintaining a vertical position (pitch=0, yaw=0). During gripper rotation a *constant* 5 N horizontal force was applied, using a combination of handle, ball bearings and calibrated weights. Force readings for this experiment are presented in Fig. 17. The results show a clear force reading dependency on gripper orientation. For -45 degrees rotation , both pitch and tip forces are close to 0, while side forces have a maximum. This is the orientation for which the applied force was *axially perpendicular* to the gripper side. The results of the previous experiment indicate that force reading during assembly might depend on gripper orientation during part pick up. It is therefore important to determine what is the best gripper grasping position in order to maximize sensor output. We consider this to be a perpendicular to the general direction P_1G_1, since this is the direction in which initial contact forces appear.

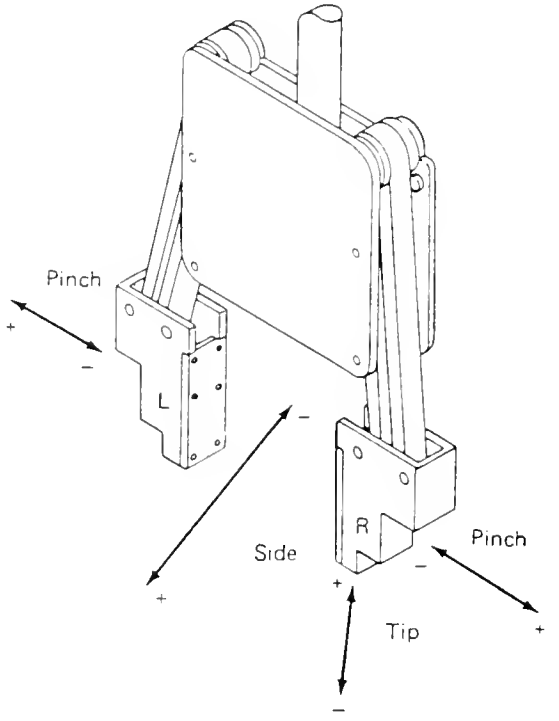


Fig. 16 IBM 7565 Force Sensor Geometry

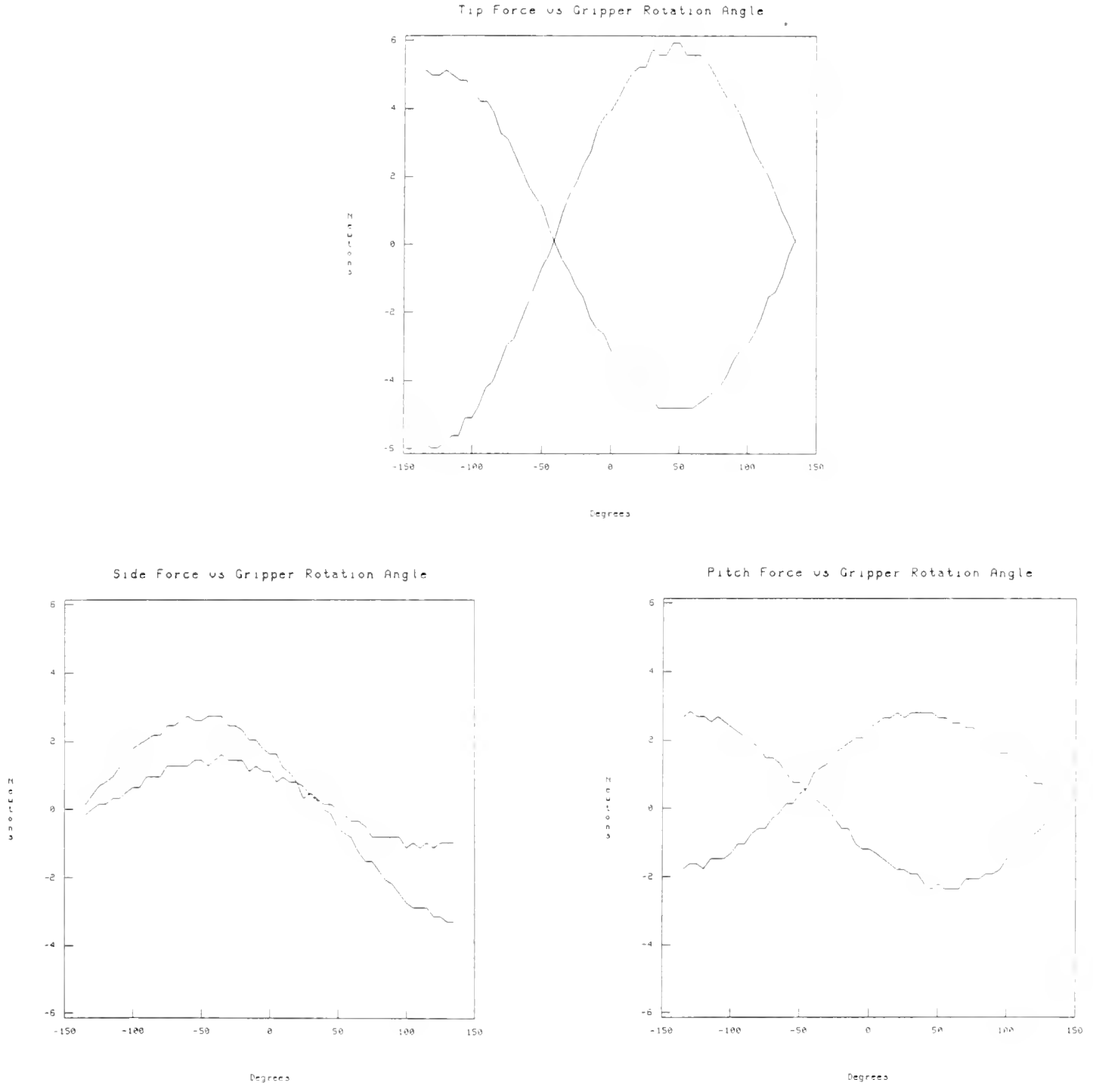


Fig. 17 Force readings vs gripper orientation

Force readings during assembly are presented in Fig. 18 to 26. This readings represent *processed data* that has been compensated for sensor offset and calibration amplification. Measured force envelope during piece nesting is of the order of 5 N, which is slightly larger than anticipated. This might be due to the mechanical amplification effect of the handle-gripper combination. Tests showed a mechanical amplification of about *50%* which means true contact forces of about 2.5 N. This force level meets conditions set in Chapter 2, which explains why no slip is present during initial assembly steps.

Force readings are further influenced by cross coupling effects due to sensor design. Tests have shown that cross coupling accounts for about *5 to 10%* residual force readings, as presented in Fig. 7. Second order dynamic effects have not been considered here since assembly is done at very low speed (0.2 inch/second).

Fig. 18 to 20 present force data for nesting only, with no rotation error. Fig. 21 to 23 are test data for puzzle piece nesting with 4.63 degrees rotation error. Finally Fig. 24 to 26 show forces during a complete assembly including fine motion moves. Here nesting is only the initial part of the graphs, where force patterns detailed in Fig. 18 to 20 are recognizable. During fine assembly forces occasionally pass the limit of magnetic pull resistance, and slip is present.

When analyzing Fig. 18 we note a large tip force encrease from *A* to *B*, followed by a sharp decrease *C* to *D*. These represent the end of raw positioning moves at point B and the detection of edge 1 at point D, as described in Chapter 4. Both sensor outputs follow the same pattern *ABCDE*, but separate along *EFG* and *EHI* respectively. This corresponds to nesting moves that detect edge 3 and 4. Here both sensors are subject to torques in opposing direction which explains the mirror symmetry of *EFG* versus *EHI*. The same general pattern is followed by side and pitch forces. Fig. 20 details the same nesting moves, but here force variation corresponding to detection of edge 1, 2, 3 and 4 is less distinct compared to Fig. 18. These results reflect the fact that during the test, initial contact was already on edge 4, and the assembly algorithm did not succeed in nesting the 2 puzzle pieces. It is therefore important to estimate the limit on rotational and translational errors that the assembly algorithm can tolerate.

6. Assembly robustness

The vision solution given as input to the robot assembly algorithm may contain both rotation and translation errors. We define the vision rotation error θ_{err} as given by (6.1).

$$\theta_{err} = \theta' - \theta \tag{6.1}$$

θ_{err} = rotation error from vision solution
θ' = solution returner by vision match
θ = correct solution

Due to puzzle asymmetry, any rotation errors produce translation errors of the initial contact point at the end of raw positioning. This translation errors are given by (6.2), (6.3).

$$X_{J'} - X_J = G_{3xrobot'} - G_{3xrobot} + 2(T+l)sin(\frac{\theta_{err}}{2})cos(\frac{2\beta+\theta_{err}}{2}) \quad (6.2)$$

$$Y_{J'} - Y_j = G_{3yrobot'} - G_{3yrobot} + 2(T+l)sin(\frac{\theta_{err}}{2})sin(\frac{2\beta+\theta_{err}}{2}) \quad (6.3)$$

$X, Y_{J'}$ = robot coordinates of contact point with vision rotation error
X, Y_J = robot coordinates of contact point without rotation errors
θ_{err} = rotation error from vision
$T = a - cos(\theta)a + (b+f)sin(\theta)$

By applying (6.2), (6.3) to the specifics of Piece 1 and Piece 2, these equations become (6.4), (6.5).

$$X_{J'} - X_J = 3.71sin(\theta_{err}) - 0.011sin^2(\frac{\theta_{err}}{2}) \quad (6.4)$$

$$Y_{J'} - Y_J = -0.005sin(\theta_{err}) + 4.36sin^2(\frac{\theta_{err}}{2}) \quad (6.5)$$

The translation error from vision may not be larger than the pseudo radius of the stationary puzzle concavity, or the peg will not intersect the concavity. The other condition requires the gripper to be as close as possible to the perpendicular to P_1G_1, as discussed in Chapter 5. An estimate for force sensor errors due to lack of proper positioning requires future work, while we presently address the geometrical condition only. For a pseudo radius of 0.25 inches, the condition set in (6.5) returns a maximum θ_{err} of 4 degrees for successful assembly. Tests have been run with rotation errors varying between 0 and 4.6 degrees. Successful nesting was obtained at initial rotation errors of 3 degrees, while the test failed at 4.6 degrees, results that confirm the theoretical prediction. The critical step of rotation error elimination had a success rate of about 50% mainly due to lack of force sensor sensitivity. In order to improve assembly robustness it may be necessary to modify the nesting algorithm to take alternative actions in case edge 1 was not detected. The algorithm might reverse the scan on a perpendicular direction to P_1G_1 to detect edge 3 and 4 and *then* detect edge 1 and 2. We want to

stress again that rotational errors influence both the geometrical and force sensing aspects of assembly. The larger the rotation error, the more different the sensors response, which is due to sensor asymmetry. This is evident when comparing Fig. 18 and 21.

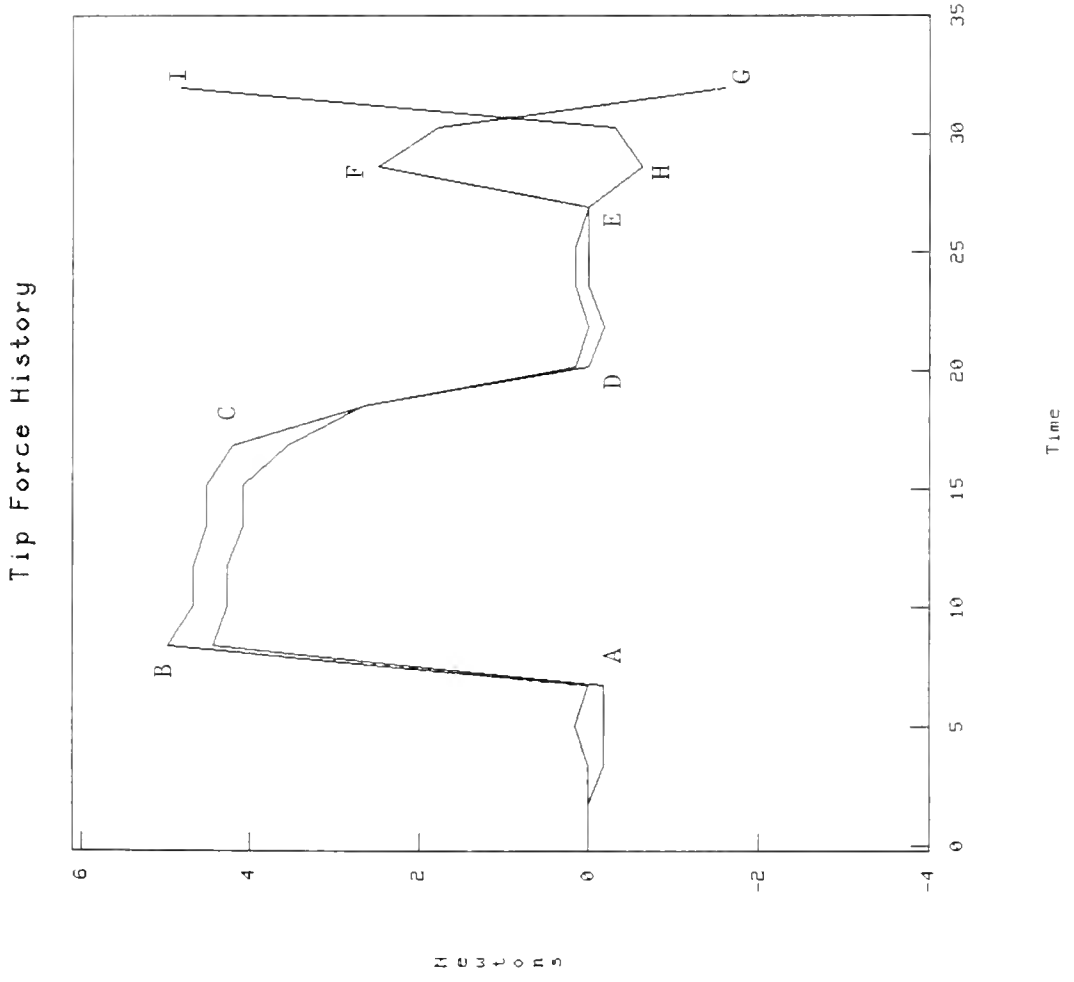


Fig. 18 Force sensor output for puzzle piece "nesting" with no rotation error (tip force)

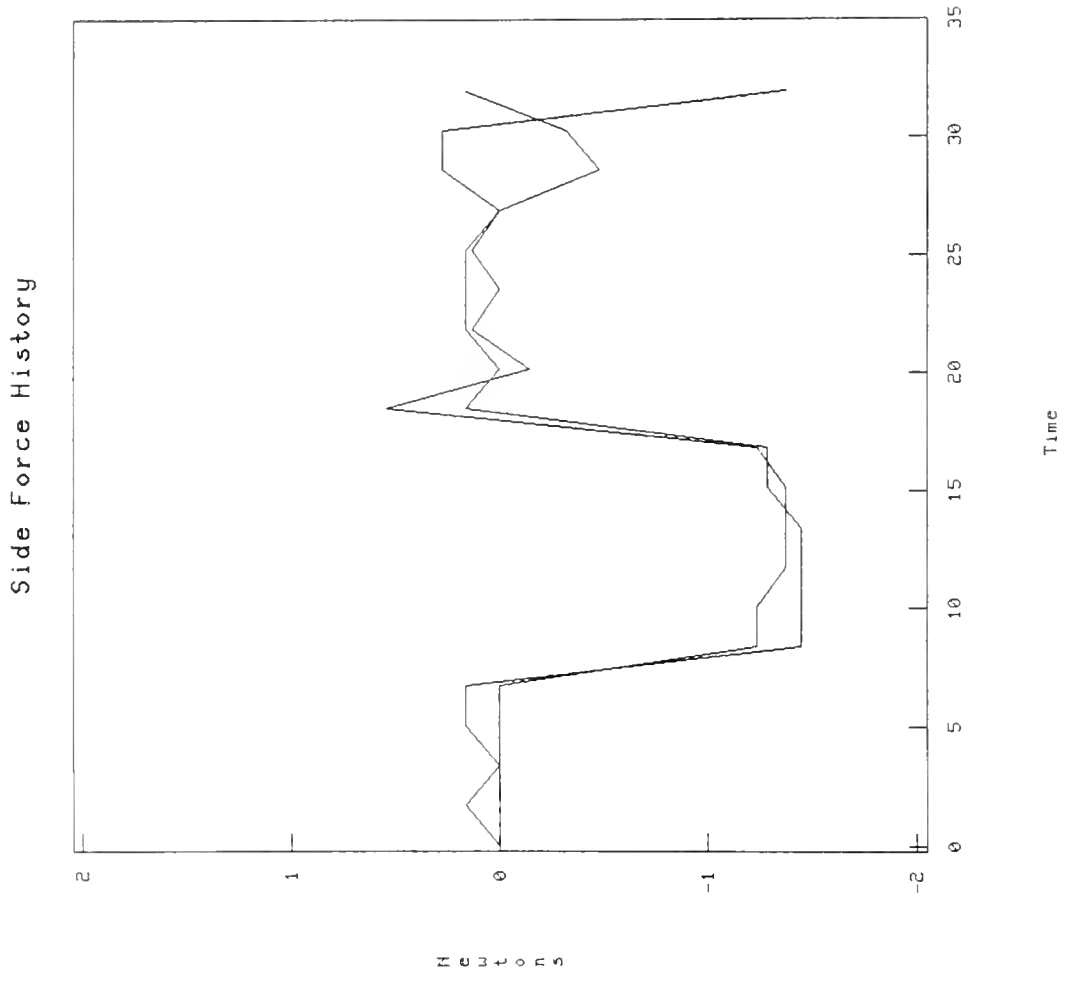


Fig. 19 Force sensor output for puzzle piece "nesting" with no rotation error (side force)

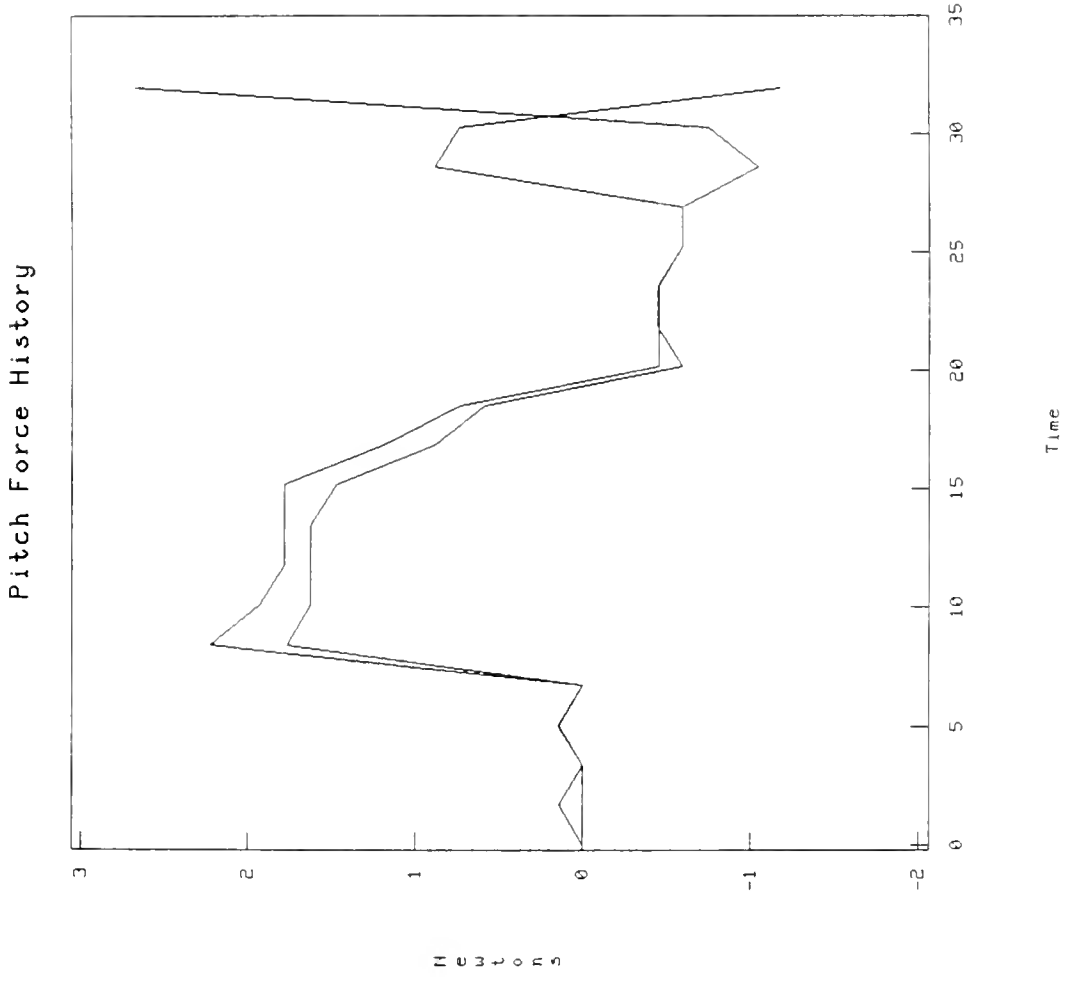


Fig. 20 Force sensor output for puzzle piece "nesting" with no rotation error (pitch force)

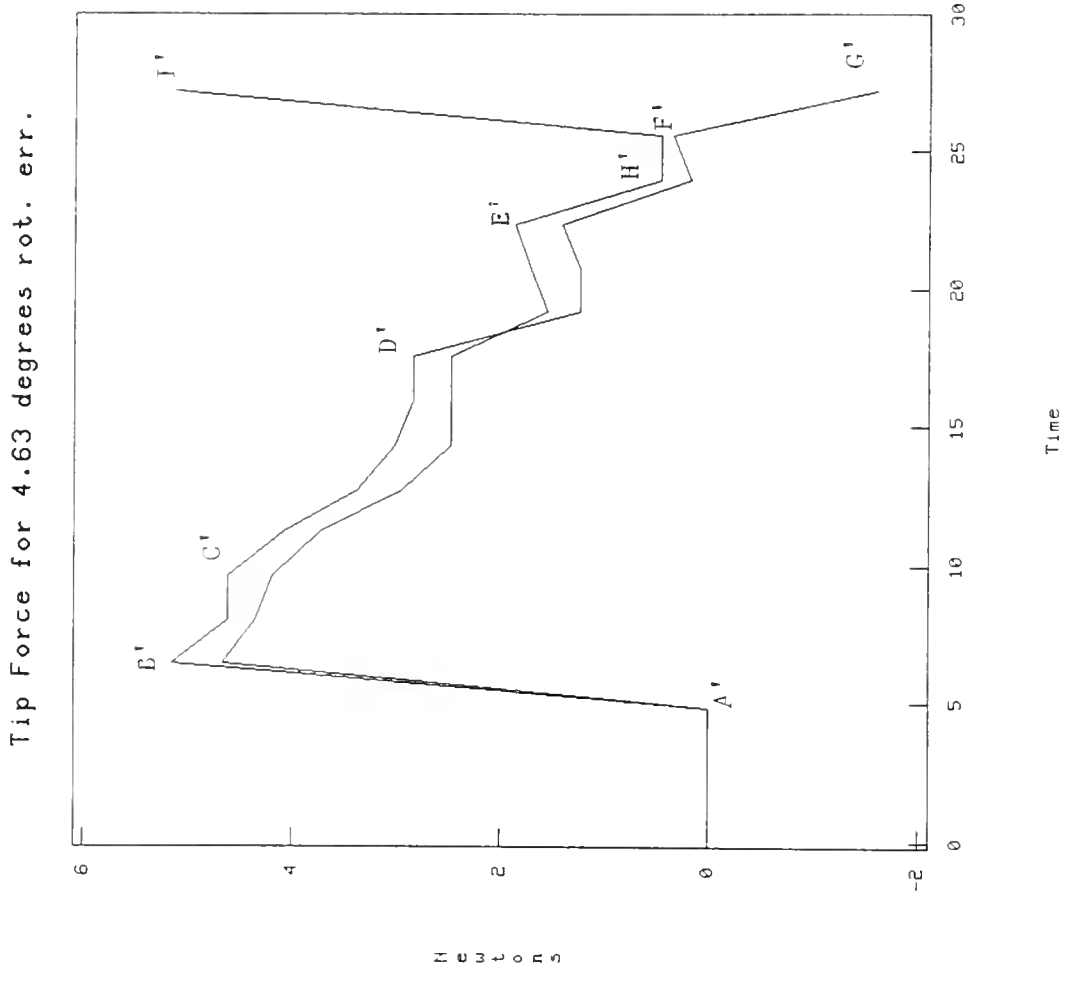


Fig. 21 Force readings for "nesting" with 4.63 degrees rotation error (tip force)

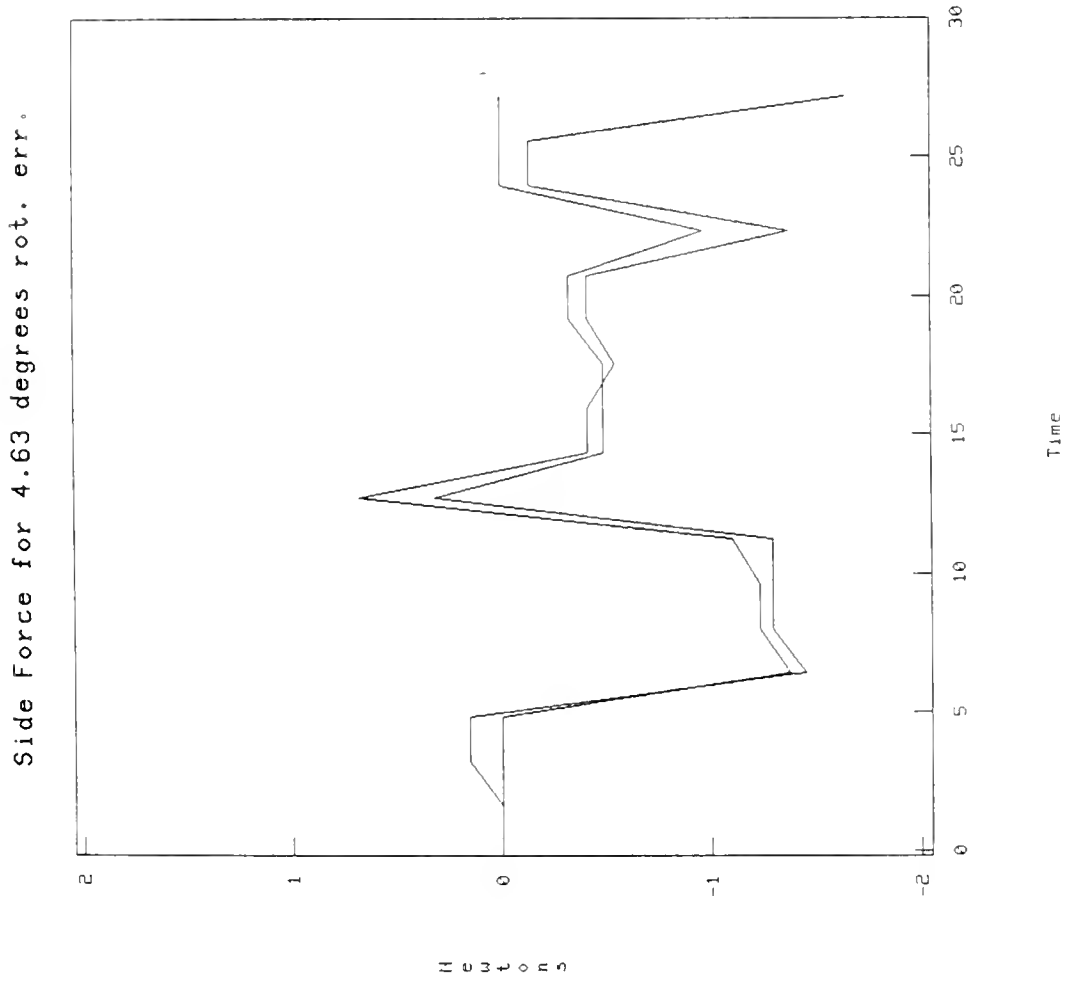


Fig. 22 Force readings for "nesting" with 4.63 degrees rotation error (side force)

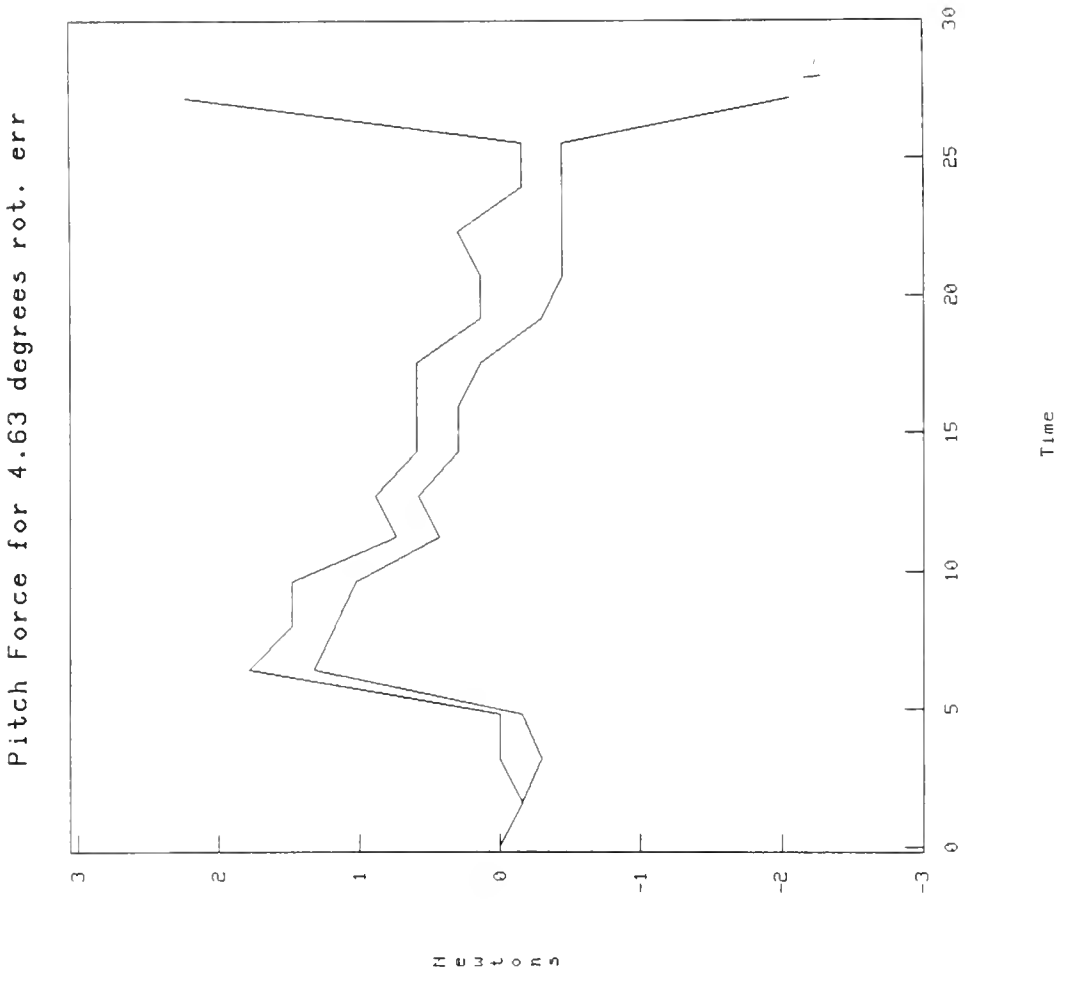


Fig. 23 Force readings for "nesting" with 4.63 degrees rotation error (pitch force)

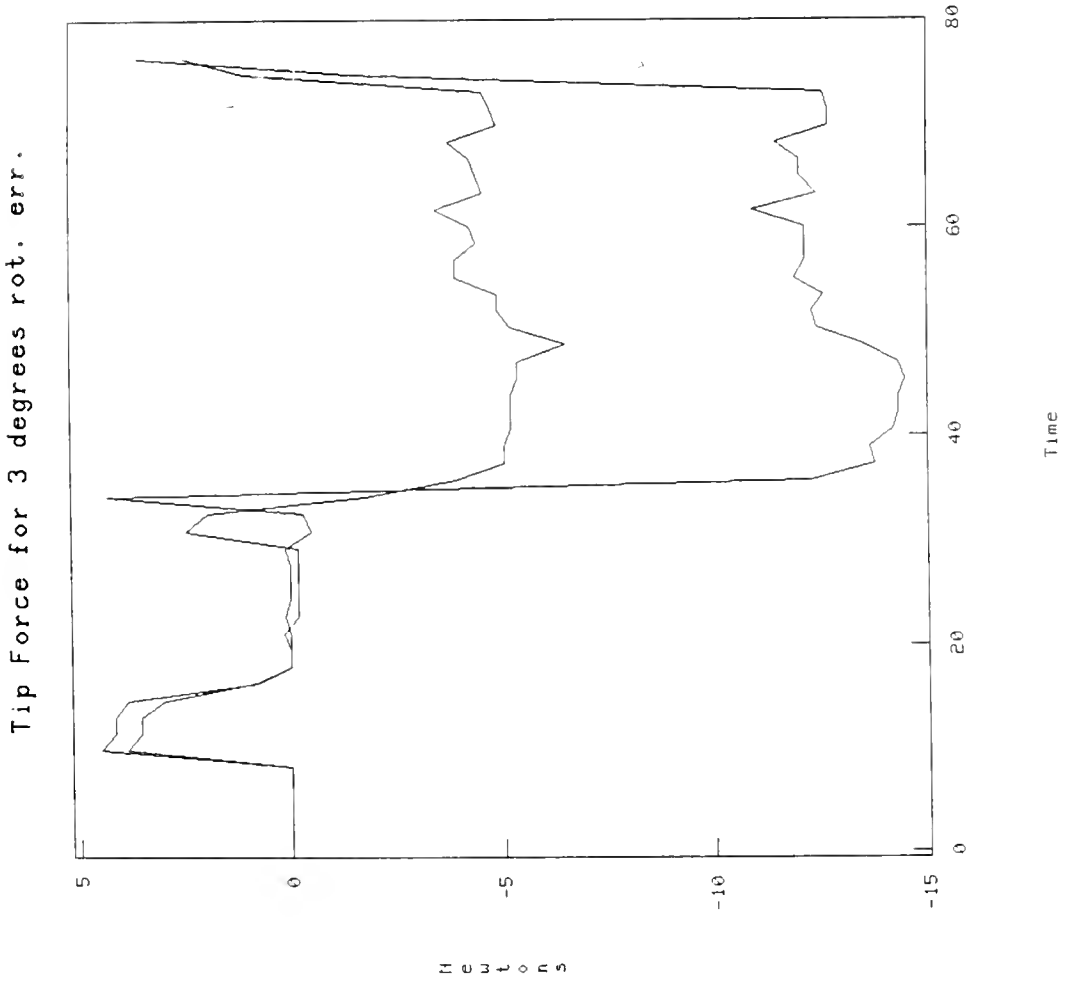


Fig. 24 Forces for complete assembly with 3 degree rotation error (tip force)

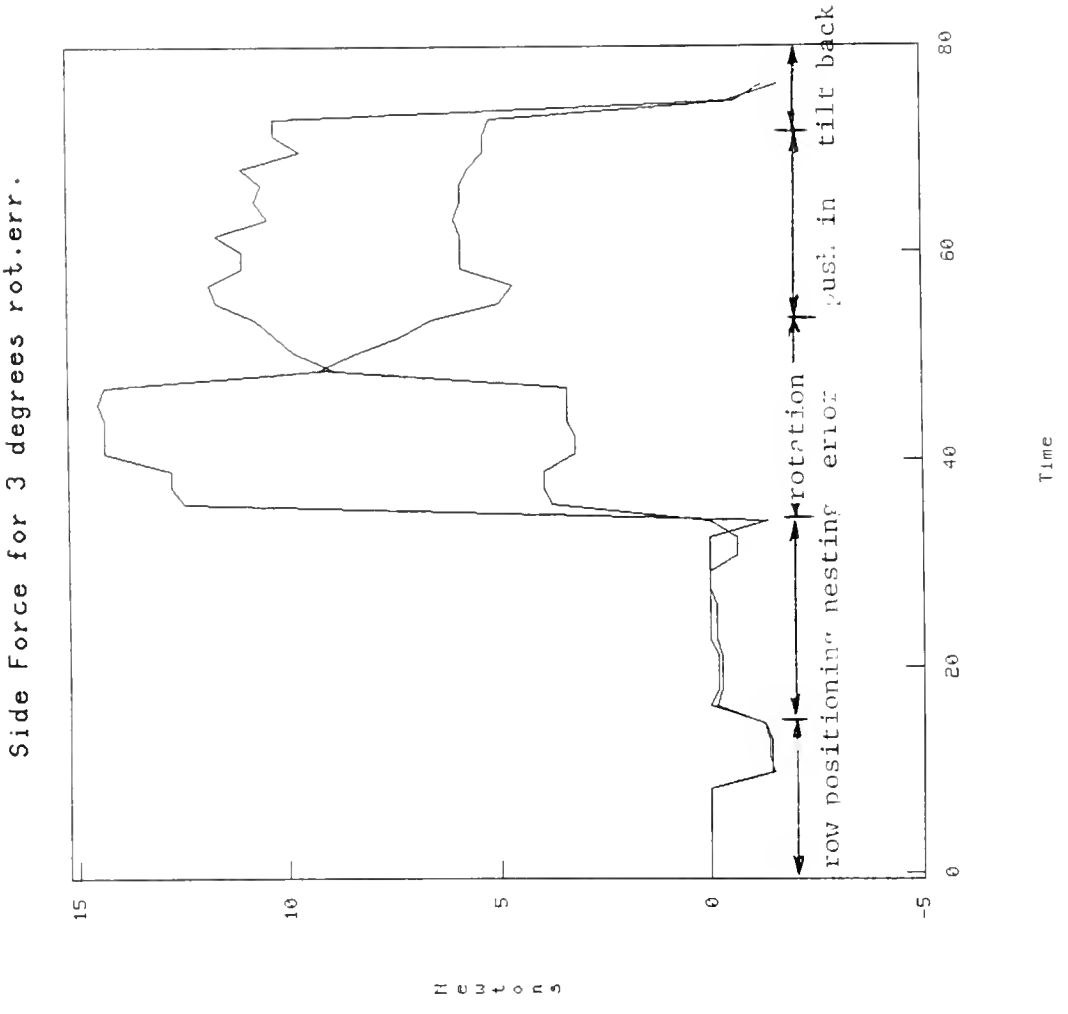


Fig. 25 Force for complete assembly with 3 degree rotation error (side force)

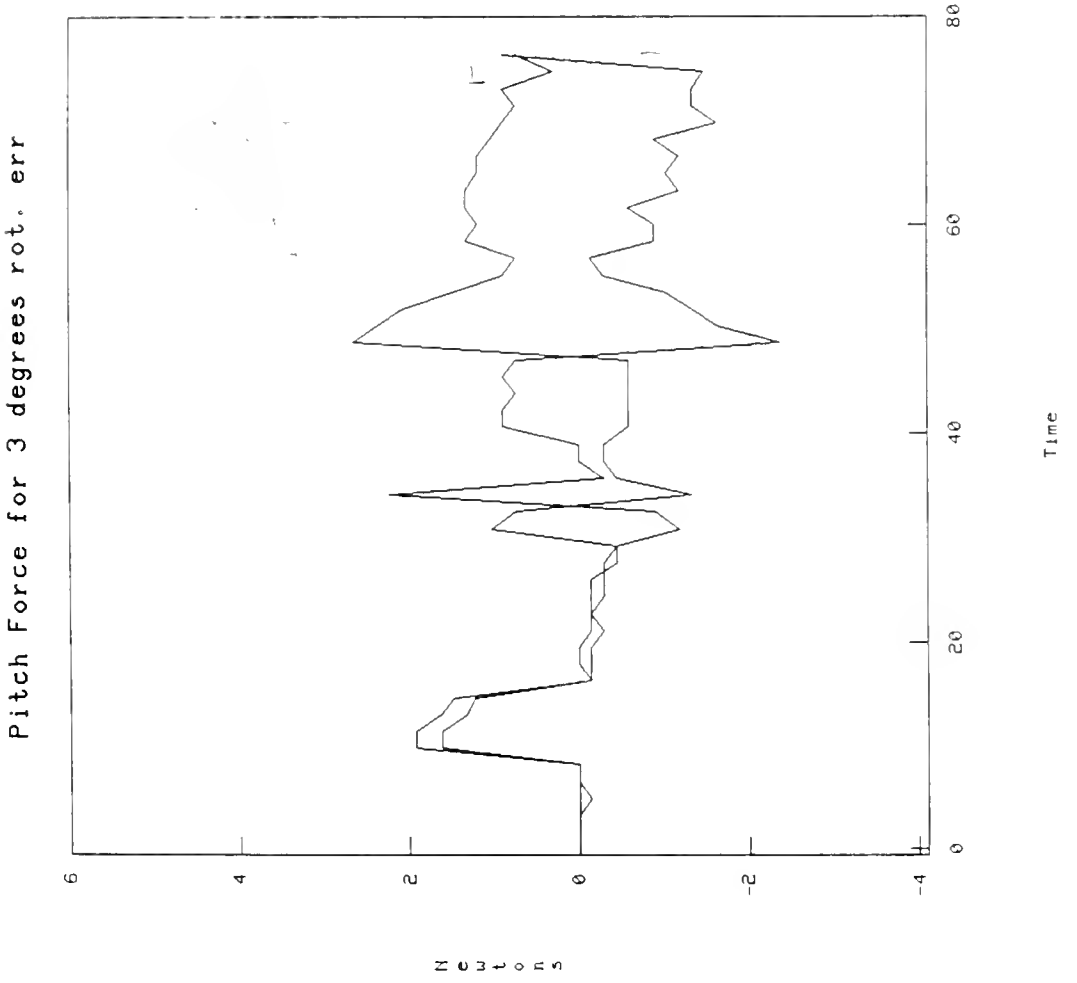


Fig. 26 Force for complete assembly with 3 degree rotation error (pitch force)

References

1. Thomas Lozano-Perez , "Automatic Synthesis of Fine-Motion Strategies for Robots," *The International Journal of Robotics Research* 3M.I.T, (1984).
2. , *Foroseal Flexible Magnetic Sheet and Strip Application Guide,* BFGoodrich Co., Akron, OH (1982).
3. Steven Buckley, "," Planning and Teaching Compliant Motion Strategies, M.I.T., Boston (1987). Ph. D. Thesis
4. Grigore C. Burdea and Haim Wolfson, "," Automated Assembly of a Jig-Saw Puzzle Using the I.B.M. 7565 Robot, New York University, New York (1985). Technical Report 188
5. Grigore C. Burdea, "," Complex System and Tool for Fine Robot Assembly, New York University, Robotics Laboratory, New York (September 1986). Technical Report 276
6. Hirochika Inoue, "," Force Feedback in Precise Assembly Tasks, M.I.T., Boston (1974). M.I.T. A.I.M.-308
7. Alan Kalvin, Edith Schonberg, Jacob Schwartz, and Micha Sharir, "," Two Dimensional Model Based Boundary Matching Using Footprints, New York University, Robotics Laboratory, New York (1985). Technical Report 162
8. Thomas Lozano-Perez, "Automatic Planning of Manipulator Transfer Movements," *Robot Motion, Planning and Control,* pp. 498-535 M.I.T. Press, (1985).
9. Donald S. Seltzer, "Tactile Sensory Feedback for Difficult Robot Tasks," *Robot 6 Conference*, S.M.E., (1982). Technical report MS82-220
10. Sergio Simmunovic, "," An Informational Approach to Parts Mating, M.I.T, Boston (1979). Ph. D. Thesis
11. P. Will and D. Grossman, "An Experimental System For Computer-Controlled Mechanical Assembly," *IEEE Transactions on Computers* **24**(1975).
12. Haim Wolfson, Edith Schonberg, Alan Kalvin, and J. Lamdan, "Solving Jig-Saw Puzzle by Computer," *Approaches to Intelligent Decision Support* **Annals of Operations Research**(1987).

```
#include <stdio.h>
#include <vsh.h>
        analyze(true, cgx, cgy)  /* analyze the image and get cg piece if "found" */        analyze
        int *true, *cgx, *cgy;
        {
                short buf[512];
                int i, j, imd,left,leftmost,right,rightmost;
                int upmost, downmost, found;

                found=0;  /*start with assumption that no pieces are present                10
                open image file */
                imd = iopen("temp.img",0);  /* VICOM command use cc .. -lv */
                printf("analyze image file\n");
                j=1,
tryagain:       while(j<129)    /* only 128 rows in memory */
                {
                ireads(imd,buf,512); /*read one line at a time */
                        /*printf("row %d read\n", j); */
                        for(i=30;i<483&&buf[i]==0.0;i=i+4);
                        /*printf(" i= %d\n", i); */                                        20
                        if ((i<483)&&(buf[i]<0.0))
                        {
/* big loop to determine if object was found */
                                leftmost=i;
                                upmost=j;
                                /*printf("leftmost= %d\n", leftmost);  */

                                /*printf("upmost= %d\n", upmost);  */
                                for(;i<483&&buf[i]<0.0;i++)
                                {                                                        30
/*scan the line to see where the object ends */
                                        rightmost=i;
                                }
                                /*printf("rightmost=%d\n", rightmost); */
newline:                        j=j+1;
                                if(j<130)
                                {
/* check to see if the bottom of picture was reached */
                                ireads(imd,buf,512);
                                /*printf("line %d read\n", j); */                          40
/* read new line after first that was not empty */
                                for(i=30;i<483&&buf[i]==0.0;i=i+4);
                                if((i<483)&&(buf[i]<0.0))
                                {
                                        left=i;
                                        if(leftmost>left) leftmost=left;
                                        /*printf("newleftmost=%d\n",leftmost);
*/
/* readjust left margin */
                                        for(;i<483&&buf[i]<0.0;i++)                      50
                                        {
                                                right=i;
                                        }
```

```
                                        if(right>rightmost) rightmost=right;
                                        /*printf("newrightmost=%d\n",
                                                rightmost);*/
/* readjust right margin */
                                        downmost=j;
                                        /*printf("downmost=%d\n", downmost);*/
                                                                                60
                                        found=1;
                                        goto newline;
/*read in newline only if not junk */
                                }
                                else
                                {
/* empty line found after full line(s) */
                                        if(!found)
                                        {
                                                j=j+1;                          70
                                                goto tryagain;
/* discard results and start all over again , it was a junk point */
                                        }
                                }
                                }  /* end of for j<130 loop */
/* here when object found is not junk */
                                if(found) goto calculate;
                        } /* end of large if loop */
                        j=j+1;
                }  /* end of big while loop */                              80
calculate:      if((j>128)&&(!found))
                {
                        *true=0;
                        *cgx=0;
                        *cgy=0;
                        return;
                }
                else
                {
                        if(found)                                           90
                        {
                                *true=1;
                                *cgx=(leftmost + rightmost)/2;
                                *cgy=3*(upmost+downmost-2)/2 + 48;
                        }
                        iclose(imd);
                } /* end of else */
        } /* end of analyze */
```

```
/*

        Approach.c routine calculates the first move after robot
        has picted up piece 2.  In this calculations, data is
        provided by vision routines and match routine.

Inputs:
        g1_x,y:         center of gravity of moving piece (piece2)
        p1_x,y:         center of gravity of matching side of moving piece
        p2_x,y:         center of gravity of matching side of stationary piece
                        (piece1)                                          10
        left_x:         leftmost x on image of stationary piece
        theta:          rotation angle of matching side
        j1_x,y:         robot x,y arm coordinates when stationary piece is
                        "centered" in the camera field of view.

Outputs:
        j3_x,y:         robot arm x,y coordinates for approach prior to
                        assembly

*/                                                                        20
#include <stdio.h>
#include <math.h>
#include "transform.h"

approach(g1_x,g1_y,p1_x,p1_y,p2_x,p2_y,left_x,theta,j1_x,j1_y,j3_x,j3_y)      approach

int   g1_x,g1_y,p1_x,p1_y,p2_x,p2_y,left_x;
float theta,j1_x, j1_y, *j3_x,*j3_y;

{                                                                         30
        float  j3_xtemp, j3_ytemp;
        int    g3_x, g3_y;
/* g3_x,y:  center of gravity of moving piece for end of approaching move as
            shown on vicom image
*/

theta=theta*3.1415927/180.;
/*transform theta in radians  to be used by trigonometric library */

g3_x= (p2_x + left_x*(XSCALE-1) + 0.5)/XSCALE + sin(theta)*(g1_y - p1_y) +   40
      cos(theta)*(g1_x - p1_x)/XSCALE;
g3_y= p2_y + sin(theta)*(p1_x - g1_x)/XSCALE + cos(theta)*(g1_y - p1_y);

imgvsrobt(g3_x,g3_y, &j3_xtemp, &j3_ytemp);

*j3_x= j3_xtemp + j1_x;
*j3_y= j3_ytemp + j1_y;

printf("robot x target for approach = %f\n", *j3_x);
printf("robot y target for approach = %f\n", *j3_y);                      50

}
```

NYU COMPSCI TR-290 c.1
Burdea, Grigore C
2 piece jig-saw puzzle
robot assembly with
vision, position, and